SIEMENS

SINUMERIK

SINUMERIK 840D sl/ SINUMERIK 828D Extended Functions

Function Manual

Valid for

Control System
SINUMERIK 840D sl/840DE sl
SINUMERIK 828D

Software	*Version*
NCU system software for 840D sl/840DE sl	2.6
NCU system software for 828D	2.6

09/2009
6FC5397-1BP20-0BA0

Preface
A4: Digital and analog NCK I/Os 1
B3: Several operator panels connected to several NCUs, distributed systems - only 840D sl 2
B4: Operation via PG/PC - only 840D sl 3
H1: Manual travel and handwheel travel 4
K3: Compensation 5
K5: Mode groups, channels, axis interchange 6
M1: Kinematic transformation 7
M5: Measuring 8
N3: Software cams, position switching cycles - only 40D sl 9
N4: Punching and nibbling - only 840D sl 10
P2: Positioning axes 11
P5: Oscillation - only 840D sl 12
R2: Rotary axes 13
S3: Synchronous spindle 14
S7: Memory configuration 15
T1: Indexing axes 16
W3: Tool change 17
W4: Grinding-specific tool offset and monitoring functions - only 840D sl 18
Z2: NC/PLC interface signals 19
Appendix A

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation for the specific task, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be adhered to. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

Order number: 6FC5397-1BP20-0BA0
℗ 10/2009

Copyright © Siemens AG 2009.
Technical data subject to change

Preface

SINUMERIK® Documentation

The SINUMERIK documentation is organized in three parts:

- General documentation
- User documentation
- Manufacturer/service documentation

Information on the following topics is available at
http://www.siemens.com/motioncontrol/docu:

- Ordering documentation
  Here you can find an up-to-date overview of publications.
- Downloading documentation
  Links to more information for downloading files from Service & Support.
- Researching documentation online
  Information on DOConCD and direct access to the publications in DOConWEB.
- Compiling individual documentation on the basis of Siemens contents with the My Documentation Manager (MDM), refer to http://www.siemens.com/mdm.

  My Documentation Manager provides you with a range of features for generating your own machine documentation.
- Training and FAQs

  Information on our range of training courses and FAQs (frequently asked questions) is available via the page navigation.

Target group

This publication is intended for:

- Project engineers
- Technologists (from machine manufacturers)
- System startup engineers (Systems/Machines)
- Programmers

Benefits

The function manual describes the functions so that the target group knows them and can select them. It provides the target group with the information required to implement the functions.

Standard version

This documentation only describes the functionality of the standard version. Extensions or changes made by the machine tool manufacturer are documented by the machine tool manufacturer.

Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

Further, for the sake of simplicity, this documentation does not contain all detailed information about all types of the product and cannot cover every conceivable case of installation, operation or maintenance.

Installation

Structure of this Function Manual:

- Inner title (page 3) with the title of the Function Manual, the SINUMERIK controls as well as the software and the version for which this version of the Function Manual is applicable and the overview of the individual functional descriptions.
- Description of the functions in alphabetical order (e.g. A2, A3, B1 etc.)
- Appendix with:
  - List of abbreviations
  - Fax template for feedback for documentation
  - Overview
- Index of terms

Note

For detailed descriptions of data and alarm see:

- Machine and setting data:
  Detailed description of machine data (only electronically on DOConCD or DOConWEB)
- NC/PLC interface signals:
  Function Manual Basic Functions; NC/PLC Interface Signals (Z1)
  Function Manual Basic Functions; NC/PLC Interface Signals (Z2)
  Function Manual Special Functions; NC/PLC Interface Signals (Z3)
- Alarms:
  Diagnostics Manual

Notation of system data

The following notation is applicable for system data in this documentation:

Signal/Data	Notation	Example
NC/PLC interface signals	... NC/PLC interface signal: <signal address> (<signal name>)	When the new gear step is engaged, the following NC/PLC interface signals are set by the PLC program: DB31, ... DBX16.0-2 (actual gear stage A to C) DB31, ... DBX16.3 (gear is changed)
Machine data	... machine data: <Type><Number> <Complete Designator> (<Meaning>)	Master spindle is the spindle stored in the machine data: MD20090 $MC_SPIND_DEF_MASTER_SPIND (Position of deletion of the master spindle in the channel).
Setting Data	... Setting data: <Type><Number> <Complete Designator> (<Meaning>)	The logical master spindle is contained in the setting data: SD42800 $SC_SPIND_ASSIGN_TAB[0] (Spindle number converter)

Note

Signal address

The description of functions include as <signal address> of an NC/PLC interface signal, only the address valid for SINUMERIK 840D sl. The signal address for SINUMERIK 828D should be taken from the data lists "Signals to/from ..." at the end of the particular description of functions.

Quantity structure

Explanations concerning the NC/PLC interface are based on the absolute maximum number of sequential components:

- Mode groups (DB11)
- Channels (DB21, etc.)
- Axes/spindles (DB31, etc.)

Data types

The following elementary data types are used in the control system:

Type	Meaning	Value range
INT	Signed integers	-2147483648 ... +2147483647
REAL	Figures with decimal point acc. to IEEE	$\pm(2.2*10^{-308} \ldots 1.8*10^{+308})$
BOOL	Truth values TRUE (1) and FALSE (0)	1, 0
CHAR	ASCII characters	Corresponding to code 0 to 255
STRING	Character string, number of characters in [...]	Maximum of 200 characters (no special characters)
AXIS	Axis names only	All axis identifiers in the channel
FRAME	Geometrical parameters for moving, rotating, scaling, and mirroring	
Arrays can only be formed from similar elementary data types. Up to 3-dimensional arrays are possible.		

Technical Support

If you have any questions, please contact our hotline:

	Europe/Africa
Phone	+49 180 5050 - 222
Fax	+49 180 5050 - 223
€0.14/min. from German landlines, cell phone prices may differ	
Internet	http://www.siemens.de/automation/support-request

	Americas
Phone	+1 423 262 2522
Fax	+1 423 262 2200
E-mail	mailto:techsupport.sea@siemens.com

	Asia/Pacific
Phone	+86 1064 757575
Fax	+86 1064 747474
E-mail	mailto:support.asia.automation@siemens.com

Note

You will find telephone numbers for other countries for technical support on the Internet: http://www.automation.siemens.com/partner

Questions about the manual

If you have any queries (suggestions, corrections) in relation to this documentation, please send a fax or e-mail to the following address:

Fax: +49 9131- 98 2176
Email: mailto:docu.motioncontrol@siemens.com

A fax form is available in the appendix of this document.

SINUMERIK Internet address

http://www.siemens.com/sinumerik

SINUMERIK 828D system performance (region)

	PPU 260 / 261		PPU 280 / 281	
	T	M	T	M
System performance				
Basic quantity of axes/spindles	3	4	3	4
Max. number of axes/spindles	6	6	8	6
Max. number of interpolating axes	4	4	4	4
Max. number of channels/mode groups	1/1	1/1	1/1	1/1
Min. block change time	~ 6 ms	~ 2 ms	~ 6 ms	~ 1 ms
Speed/current control cycle	125 µs	125 µs	125 µs	125 µs
CNC user memory (buffered)	3 MB	3 MB	5 MB	5 MB
CNC functions				
Tool Management	●	●	●	●
Number of tools/cutting edges	128/256	128/256	256/512	256/512
Safety Integrated: Safe Torque Off, Safe Brake Control	●	●	●	●
Safety Integrated: Safe velocity	○	○	○	○
Max. number of ASUBs (permanently set)	2	2	2	2
TRANSMIT/TRACYL	○	○	○	○
Inclined Y axis	-	-	○	-
Synchronous spindle for counterspindle	-	-	○	-
Synchronous spindle for polygon machining	○	-	○	-
Gantry	○	○	○	○
Temperature compensation	●	●	●	●

	PPU 260 / 261		PPU 280 / 281	
	T	**M**	**T**	**M**
HMI, CNC programming				
ShopMill/ShopTurn HMI functions	●	●	●	●
DIN/ISO programming with programGUIDE	●	●	●	●
Online ISO dialect interpreter	●	●	●	●
ShopMill/ShopTurn machining step programming	○	○	○	○
Measuring cycles	○	○	○	○
Simulation in surface display	●	●	●	●
Simulation in 3D display	○	○	○	○
PLC				
SIMATIC S7-200 (integrated)	●	●	●	●
Basic I/O modules: PP 72/48D PN PP 72/48D 2/2A PN (being prepared)	○	○	○	○
PLC cycle time	6 ms	6 ms	6 ms	6 ms
Max. number of PLC operations	24000	24000	24000	24000
Servo-synchronous high-speed PLC task	●	●	●	●
Reaction time to process interrupts (terminal to terminal)	7.5 ms	7.5 ms	4.5 ms	4.5 ms
Max. number of digital inputs/outputs	288/192	288/192	360/240	360/240
Max. number of analog inputs/outputs	8/8	8/8	10/10	10/10
Commissioning functions				
Service Planner (maintenance scheduler)	●	●	●	●
Easy Extend (for managing machine units)	●	●	●	●
Easy Archive (data archiving)	●	●	●	●

T Turning
M Milling
● Standard (basic scope)
○ CNC option
\- Not available

Contents

Preface **3**

1 A4: Digital and analog NCK I/Os **27**

1.1 Brief description 27

1.2 NCK I/O via PLC 28
1.2.1 General functionality 28
1.2.2 NCK digital inputs/outputs 35
1.2.2.1 NCK digital inputs 35
1.2.2.2 NCK digital outputs 37
1.2.3 Connection and logic operations of fast NCK inputs/outputs 41
1.2.4 NCK analog inputs/outputs 43
1.2.4.1 NCK analog inputs 43
1.2.4.2 NCK analog outputs 45
1.2.5 Direct PLC I/Os, addressable from the NC 49
1.2.6 Analog-value representation of the NCK analog input/output values 52
1.2.7 Comparator inputs 54

1.3 NCK I/O via PROFIBUS - only 840D sl 58
1.3.1 Functionality 58
1.3.2 Parameter assignment 59
1.3.3 Programming 61
1.3.3.1 Communication via part programs/synchronous actions 61
1.3.3.2 Communication via compile cycles 64

1.4 Constraints 66
1.4.1 NCK I/O via PLC 66
1.4.2 NCK I/O via PROFIBUS - only 840D sl 67

1.5 Examples 68
1.5.1 NCK I/O via PLC 68
1.5.1.1 Writing to PLC-I/Os 68
1.5.1.2 Reading from PLC-I/Os 69
1.5.2 NCK I/O via PROFIBUS - only 840D sl 70
1.5.2.1 PROFIBUS-I/O in write direction 70
1.5.2.2 PROFIBUS-I/O in read direction 71
1.5.2.3 Query of the RangeIndex in case of "PROFIBUS-I/O in write direction" 73

1.6 Data lists 75
1.6.1 Machine data 75
1.6.1.1 General machine data 75
1.6.1.2 Channel-specific machine data 76
1.6.2 Setting data 76
1.6.2.1 General setting data 76
1.6.3 Signals 76
1.6.3.1 Signals to NC 76
1.6.3.2 Signals from NC 77

2 B3: Several operator panels connected to several NCUs, distributed systems - only 840D sl **79**

2.1 Brief description 79
2.1.1 Topology of distributed system configurations 79

2.1.2 Several operator panels and NCUs with control unit management (option) ... 84
2.1.2.1 General information ... 84
2.1.2.2 System features ... 85
2.1.2.3 Hardware ... 86
2.1.2.4 Functions ... 88
2.1.2.5 Configurability ... 89
2.1.3 Several operator panel fronts and NCUs, standard functionality ... 90
2.1.3.1 System features ... 90
2.1.3.2 Functions ... 91
2.1.3.3 Configurability ... 93
2.1.3.4 MPI/OPI network rules ... 97
2.1.4 NCU link ... 98
2.1.4.1 General information ... 98
2.1.4.2 Types of distributed machines ... 99
2.1.4.3 Link axes ... 101
2.1.4.4 Flexible configuration ... 102
2.1.4.5 User communication across the NCUs ... 103
2.1.4.6 Lead link axes ... 104
2.1.4.7 NCU link with different interpolation cycles ... 104

2.2 Several operator panel fronts and NCUs with control unit management option ... 107
2.2.1 Hardware structure ... 107
2.2.2 Properties ... 107
2.2.3 Configuration file NETNAMES.INI ... 109
2.2.4 Structure of the configuration file ... 109
2.2.5 Creating and using the configuration file ... 114
2.2.6 Power up ... 115
2.2.7 HMI switchover ... 118
2.2.8 Suppression ... 118
2.2.9 Connection and switchover conditions ... 120
2.2.10 Implementation of control unit switchover ... 121
2.2.11 Operator interface ... 121
2.2.12 Operating mode switchover ... 122
2.2.13 MCP switchover ... 124
2.2.14 PLC program "Control Unit Switchover" ... 125

2.3 Several operator panel fronts and NCUs, standard functionality ... 130
2.3.1 Configurations ... 131
2.3.2 Switchover of connection to another NCU ... 135
2.3.3 Creating and using the configuration file ... 135
2.3.4 Power up ... 136
2.3.5 NCU replacement ... 137

2.4 Restrictions for switchover of operator components ... 138

2.5 NCU link ... 139
2.5.1 Introduction ... 139
2.5.2 Technological description ... 140

2.6 Link axes ... 142
2.6.1 Configuration of link axes and container axes ... 144
2.6.2 Axis data and signals ... 148
2.6.3 Output of predefined auxiliary functions in the case of an NCU link ... 151
2.6.4 Supplementary conditions for link axes ... 152
2.6.5 Programming with channel and machine axis identifiers ... 153
2.6.6 Flexible configuration ... 154

2.7 Axis container ... 155

2.7.1 System variables for axis containers 161
2.7.2 Machining with axis container (schematic) 163
2.7.3 Axis container behavior after Power ON 164
2.7.4 Axis container response to mode switchover 164
2.7.5 Axis container behavior in relation to ASUBs 164
2.7.6 Axis container response to RESET 164
2.7.7 Axis container response to block searches 164
2.7.8 Supplementary conditions for axis container rotations 164

2.8 Link communication 168
2.8.1 Link variables 168
2.8.2 Reading drive data via link variables 172

2.9 Configuration of a link grouping 174

2.10 Communication in link grouping 177

2.11 Lead link axis 182
2.11.1 Programming a lead link axis 186

2.12 NCU link with different interpolation cycles 187
2.12.1 Diagram of general solution 189
2.12.2 Different position control cycles 192
2.12.3 Supplementary conditions 194
2.12.4 Activating NCU links with different interpolation cycles 195
2.12.5 Different IPO cycles, behavior at power ON, RESET, etc. 195
2.12.6 System variable with different interpolation cycles 195

2.13 Link grouping system of units 196

2.14 Supplementary conditions 197
2.14.1 Several operator panels and NCUs with control unit management option 197
2.14.2 Several operator panel fronts and NCUs, standard functionality 197
2.14.3 Link axes 198
2.14.4 Axis container 198
2.14.5 Lead link axis 198
2.14.6 NCU link with different interpolation cycles 199

2.15 Examples 199
2.15.1 Configuration file NETNAMES.INI with control unit management option 199
2.15.2 User-specific reconfiguring of PLC program control unit switchover 202
2.15.2.1 Description of operational sequences (overview) 202
2.15.2.2 Description of operational sequences (details) 203
2.15.2.3 Defined logical functions/defines 213
2.15.2.4 Graphical representation of function sequences 213
2.15.3 Configuration file NETNAMES.INI, standard functionality 221
2.15.3.1 Two operator panel fronts and one NCU 221
2.15.3.2 One operator panel front and three NCUs 221
2.15.4 Quick M:N commissioning based on examples 223
2.15.4.1 Example 1 223
2.15.4.2 Example 2 226
2.15.4.3 Example 3 232
2.15.4.4 Description of FB9 236
2.15.4.5 Example of calling FB9 238
2.15.4.6 Example of override switchover 240
2.15.4.7 Switchover between MCP and HT6 241
2.15.4.8 General Information 242
2.15.5 Link axis 244
2.15.6 Axis container coordination 245

2.15.6.1 Axis container rotation without a part program wait ... 245
2.15.6.2 Axis container rotation with an implicit part program wait ... 246
2.15.6.3 Axis container rotation by one channel only (e.g. during power up) ... 246
2.15.7 Evaluating axis container system variables ... 246
2.15.7.1 Conditional branch ... 246
2.15.7.2 Static synchronized action with $AN_AXCTSWA ... 246
2.15.7.3 Wait for certain completion of axis container rotation ... 247
2.15.8 Configuration of a multi-spindle turning machine ... 248
2.15.9 Lead link axis ... 257
2.15.9.1 Configuration ... 257
2.15.9.2 Programming ... 259
2.15.10 NCU link with different interpolation cycles ... 260
2.15.10.1 Example of eccentric turning ... 260

2.16 Data lists ... 262
2.16.1 Machine data ... 262
2.16.1.1 General machine data ... 262
2.16.1.2 Channel-specific machine data ... 262
2.16.1.3 Axis/spindle-specific machine data ... 263
2.16.2 Setting data ... 263
2.16.2.1 General setting data ... 263
2.16.2.2 Axis/spindle-specific setting data ... 263
2.16.3 Signals ... 263
2.16.3.1 Signals from NC ... 263
2.16.3.2 Signals from HMI/PLC ... 264
2.16.3.3 General online interface ... 264
2.16.3.4 Signals from axis/spindle ... 266

3 B4: Operation via PG/PC - only 840D sl ... 267

3.1 Brief description ... 267

3.2 Software installation ... 269
3.2.1 System requirements ... 269
3.2.2 Installation ... 270
3.2.3 Supplementary software conditions ... 275
3.2.4 Start program ... 275
3.2.5 Close program ... 276

3.3 Operation via PG/PC ... 277
3.3.1 General operation ... 277
3.3.2 Additional information ... 279
3.3.3 Operation of operator panel fronts ... 280

3.4 Simulation of part programs ... 281

3.5 Marginal conditions ... 281

3.6 Data lists ... 281

4 H1: Manual travel and handwheel travel ... 283

4.1 Brief description ... 283
4.1.1 Overview ... 283
4.1.2 General characteristics of manual travel in JOG ... 284
4.1.3 Control of manual-travel functions via PLC interface ... 287
4.1.4 Control-system response to power ON, mode change, RESET, block search, REPOS ... 288

4.2 Continuous travel ... 289
4.2.1 General functionality ... 289
4.2.2 Distinction between inching mode continuous mode ... 289

4.2.3 Special features of continuous travel 291

4.3 Incremental travel (INC) 292
4.3.1 General functionality 292
4.3.2 Distinction between inching mode and continuous mode 292
4.3.3 Special features of incremental travel 294

4.4 Handwheel travel in JOG 295
4.4.1 General functionality 295
4.4.2 Travel request 299
4.4.3 Double use of the handwheel 303

4.5 Handwheel override in automatic mode 305
4.5.1 General functionality 305
4.5.2 Programming and activating handwheel override 309
4.5.3 Special features of handwheel override in automatic mode 311

4.6 Contour handwheel/path input using handwheel (option) 312

4.7 DRF offset 315

4.8 Start-up: Handwheels 318
4.8.1 General information 318
4.8.2 Connection via PPU - only 828D 319
4.8.3 Connected via PROFIBUS - only 840D sl 320
4.8.4 Connected via Ethernet - only 840D sl 322

4.9 Special features of manual travel 326
4.9.1 Geometry-axis manual travel 326
4.9.2 Special features of spindle manual travel 327
4.9.3 Monitoring functions 328
4.9.4 Other 330

4.10 Approaching a fixed point in JOG 331
4.10.1 Introduction 331
4.10.2 Functionality 332
4.10.3 Parameter setting 334
4.10.4 Programming 335
4.10.5 Supplementary conditions 335
4.10.6 Application example 336

4.11 Data lists 337
4.11.1 Machine data 337
4.11.1.1 General machine data 337
4.11.1.2 Channel-specific machine data 337
4.11.1.3 Axis/spindle-specific machine data 338
4.11.2 Setting data 338
4.11.2.1 General setting data 338
4.11.3 Signals 339
4.11.3.1 Signals from NC 339
4.11.3.2 Signals to mode group 339
4.11.3.3 Signals from mode group 339
4.11.3.4 Signals to channel 339
4.11.3.5 Signals from channel 340
4.11.3.6 Signals to axis/spindle 340
4.11.3.7 Signals from axis/spindle 341

5 K3: Compensation 343

5.1 Introduction 343

5.2 Temperature compensation ... 345
5.2.1 Description of functions ... 345
5.2.2 Commissioning ... 348
5.2.2.1 Temperature-dependent parameters ... 348
5.2.2.2 Temperature compensation type and activation ... 348
5.2.2.3 Maximum compensation value per IPO clock cycle ... 349
5.2.3 Example ... 349
5.2.3.1 Commissioning the temperature compensation for the Z axis of a lathe ... 349

5.3 Backlash compensation ... 352
5.3.1 Description of functions ... 352
5.3.2 Commissioning ... 353
5.3.2.1 Backlash ... 353
5.3.2.2 Weighting factor for backlash ... 354
5.3.2.3 Applying the backlash compensation step-by-step ... 354

5.4 Interpolatory compensation ... 355
5.4.1 General properties ... 355
5.4.2 Compensation of leadscrew errors and measuring system errors ... 358
5.4.2.1 Description of functions ... 358
5.4.2.2 Commissioning ... 360
5.4.2.3 Example ... 363
5.4.3 Compensation of sag and angularity errors ... 364
5.4.3.1 Description of functions ... 364
5.4.3.2 Commissioning ... 368
5.4.3.3 Examples ... 372
5.4.4 Direction-dependent leadscrew error compensation ... 381
5.4.4.1 Description of functions ... 381
5.4.4.2 Commissioning ... 382
5.4.4.3 Example ... 386
5.4.5 Extension of the sag compensation with NCU link - only 840D sl ... 389
5.4.6 Special features of interpolatory compensation ... 397

5.5 Dynamic feedforward control (following error compensation) ... 399
5.5.1 General properties ... 399
5.5.2 Speed feedforward control ... 401
5.5.3 Torque feedforward control - only 840D sl (option) ... 403
5.5.4 Dynamic response adaptation ... 405
5.5.5 Forward feed control for command- and PLC axes ... 406
5.5.6 Secondary conditions ... 408

5.6 Friction compensation (quadrant error compensation) ... 409
5.6.1 General properties ... 409
5.6.2 Conventional friction compensation ... 410
5.6.2.1 Conventional friction compensation ... 410
5.6.2.2 Commissioning of conventional friction compensation ... 412
5.6.3 Quadrant error compensation using neural networks - only 840D sl ... 421
5.6.3.1 Fundamentals ... 421
5.6.3.2 Parameterization of neural QEC ... 423
5.6.3.3 Learning the neural network ... 429
5.6.3.4 Commissioning of neural QEC ... 434
5.6.3.5 Further optimization and intervention options ... 437
5.6.3.6 Quick commissioning ... 443

5.7 Circularity test ... 446

5.8 Measures for hanging (suspended axes) ... 451
5.8.1 Electronic counterweight ... 451

5.8.2 Reboot delay 452

5.9 Data lists 455
5.9.1 Machine data 455
5.9.1.1 General machine data 455
5.9.1.2 Channel-specific machine data 455
5.9.1.3 Axis/Spindle-specific machine data 455
5.9.2 Setting data 456
5.9.2.1 General setting data 456
5.9.2.2 Axis/spindle-specific setting data 457
5.9.3 Signals 457
5.9.3.1 Signals from NC 457
5.9.3.2 Signals from mode group 457
5.9.3.3 Signals from channel 457
5.9.3.4 Signals from axis/spindle 457

6 K5: Mode groups, channels, axis interchange 459

6.1 Brief description 459

6.2 Mode groups - only 840D sl 461

6.3 Channels - only 840D sl 462
6.3.1 Channel synchronization (program coordination) 462
6.3.2 Conditional wait in continuous path mode WAITMC 465

6.4 Axis/spindle replacement 470
6.4.1 Introduction 470
6.4.2 Example of an axis replacement 472
6.4.3 Axis replacement options 473
6.4.4 Replacement behavior NC program 475
6.4.5 Axis transfer to neutral state (release) 476
6.4.6 Transferring axis or spindle in the part program 477
6.4.7 Automatic axis replacement 478
6.4.8 Axis replacement via PLC 480
6.4.9 Set axis replacement behavior variable. 483
6.4.10 Axis replacement via axis container rotation 484
6.4.11 Axis replacement with and without preprocessing stop 485
6.4.12 Exclusively PLC-controlled axis and permanently-assigned PLC axis 486
6.4.13 Geometry axis in rotated frame and axis replacement 488
6.4.14 Axis replacement from synchronized actions 489

6.5 Marginal conditions 491

6.6 Data lists 493
6.6.1 Machine data 493
6.6.1.1 General machine data 493
6.6.1.2 Channel-specific machine data 493
6.6.1.3 Axis/spindle-specific machine data 495
6.6.2 Setting data 496
6.6.2.1 Channel-specific setting data 496
6.6.3 Signals 496
6.6.3.1 Signals to/from mode group 496
6.6.3.2 Signals to/from channel 496

7 M1: Kinematic transformation 497

7.1 Brief description 497
7.1.1 TRANSMIT (option) 497
7.1.2 TRACYL (option) 498

7.1.3 TRAANG (option) ... 500
7.1.4 Chained transformations ... 500
7.1.5 Activating transformation machine data via parts program/softkey ... 501

7.2 TRANSMIT (option) ... 502
7.2.1 Preconditions for TRANSMIT ... 503
7.2.2 Settings specific to TRANSMIT ... 506
7.2.3 Activation of TRANSMIT ... 510
7.2.4 Deactivation of the TRANSMIT function ... 510
7.2.5 Special system reactions with TRANSMIT ... 510
7.2.6 Machining options for TRANSMIT ... 514
7.2.7 Working area limitations ... 520
7.2.8 Overlaid motions with TRANSMIT ... 521
7.2.9 Monitoring of rotary axis rotations over 360º ... 521
7.2.10 Constraints ... 522

7.3 TRACYL (option) ... 524
7.3.1 Preconditions for TRACYL ... 526
7.3.2 Settings specific to TRACYL ... 530
7.3.3 Activation of TRACYL ... 534
7.3.4 Deactivation of the TRACYL function ... 535
7.3.5 Special system reactions with TRACYL ... 535
7.3.6 Jog ... 538

7.4 TRAANG (option) ... 539
7.4.1 Preconditions for TRAANG (inclined axis) ... 540
7.4.2 Settings specific to TRAANG ... 543
7.4.3 Activation of TRAANG ... 546
7.4.4 Deactivation of TRAANG ... 546
7.4.5 Special system reactions with TRAANG ... 547
7.4.6 Inclined axis programming (G05, G07) ... 548

7.5 Chained transformations ... 550
7.5.1 Activating chained transformations ... 552
7.5.2 Switching off a chained transformation ... 552
7.5.3 Special characteristics of chained transformations ... 552
7.5.4 Persistent transformation ... 553
7.5.5 Axis positions in the transformation chain ... 558

7.6 Cartesian PTP travel ... 561
7.6.1 Programming of position ... 564
7.6.2 Overlap areas of axis angles ... 565
7.6.3 Examples of ambiguities of position ... 566
7.6.4 Example of ambiguity in rotary axis position ... 567
7.6.5 PTP/CP switchover in JOG mode ... 567

7.7 Cartesian manual travel (optional) ... 568

7.8 Activating transformation machine data via parts program/softkey ... 577
7.8.1 Functionality ... 577
7.8.2 Constraints ... 578
7.8.3 Control response to power ON, mode change, RESET, block search, REPOS ... 579
7.8.4 List of machine data affected ... 580

7.9 Constraints ... 583
7.9.1 Chained transformations ... 583

7.10 Examples ... 583
7.10.1 TRANSMIT ... 583
7.10.2 TRACYL ... 585

7.10.3 TRAANG 590
7.10.4 Chained transformations 591
7.10.5 Activating transformation MD via a parts program 595
7.10.6 Axis positions in the transformation chain 596

7.11 Data lists 599
7.11.1 Machine data 599
7.11.1.1 TRANSMIT 599
7.11.1.2 TRACYL 600
7.11.1.3 TRAANG 602
7.11.1.4 Chained transformations 603
7.11.1.5 Non transformation-specific machine data 603
7.11.2 Signals 603
7.11.2.1 Signals from channel 603

8 M5: Measuring 605

8.1 Brief description 605

8.2 Hardware requirements 606
8.2.1 Probes that can be used 606
8.2.2 Measuring probe connection 608

8.3 Channel-specific measuring 613
8.3.1 Measuring mode 613
8.3.2 Measurement results 614

8.4 Setting zeros, workpiece measuring and tool measuring 615
8.4.1 Preset actual value memory and scratching 615
8.4.2 Workpiece measuring 616
8.4.2.1 Input values 616
8.4.2.2 Measurement selection 623
8.4.2.3 Output values 624
8.4.2.4 Calculation method 625
8.4.2.5 Units of measurement and measurement variables for the calculation 628
8.4.2.6 Diagnostics 629
8.4.3 Types of workpiece measurement 630
8.4.3.1 Measurement of an edge (measurement type 1, 2, 3) 630
8.4.3.2 Measurement of an angle (measurement type 4, 5, 6, 7) 634
8.4.3.3 Measurement of a hole (measurement type 8) 637
8.4.3.4 Measurement of a shaft (measurement type 9) 640
8.4.3.5 Measurement of a groove (measurement type 12) 641
8.4.3.6 Measurement of a web (measurement type 13) 644
8.4.3.7 Measurement of geo axes and special axes (measurement type 14, 15) 645
8.4.3.8 Measurement of an oblique edge (measurement type 16) 647
8.4.3.9 Measurement of an oblique angle in a plane (measurement type 17) 649
8.4.3.10 Redefine measurement around a WCS reference frame (measurement type 18) 652
8.4.3.11 Measurement of a 1-, 2- and 3-dimensional setpoint selection (measurement type 19, 20, 21) 655
8.4.3.12 Measurement of an oblique angle (measurement type 24) 659
8.4.3.13 Measurement of a rectangle (measurement type 25) 663
8.4.3.14 Measurement for saving data management frames (measurement type 26) 664
8.4.3.15 Measurement for restoring backed-up data management frames (measurement type 27) 665
8.4.3.16 Measurement for defining an additive rotation for taper turning (measurement type 28) 666
8.4.4 Tool measuring 667
8.4.5 Types of workpiece measurement 668
8.4.5.1 Measurement of tool lengths (measurement type 10) 668
8.4.5.2 Measurement of tool diameter (measurement type 11) 670
8.4.5.3 Measurement of tool lengths with zoom-in function (measurement type 22) 671

8.4.5.4 Measuring a tool length with stored or current position (measurement type 23) ... 672
8.4.5.5 Measurement of a tool length of two tools with orientation ... 673

8.5 Measurement accuracy and functional testing ... 685
8.5.1 Measurement accuracy ... 685
8.5.2 Probe functional testing ... 685

8.6 Examples - only 840D sl ... 686
8.6.1 Measuring mode 1 ... 686
8.6.2 Measuring mode 2 ... 687
8.6.3 Continuous measurement ... 687
8.6.3.1 Continous measurement on completion of programmed traversing motion ... 687
8.6.3.2 Continuous measurements with deletion of distance-to-go ... 688
8.6.3.3 Continuous measurements modally over several blocks ... 688
8.6.4 Functional test and repeat accuracy ... 689

8.7 Data lists ... 691
8.7.1 Machine data ... 691
8.7.1.1 General machine data ... 691
8.7.1.2 Channel-specific machine data ... 691
8.7.2 System variables ... 691

9 N3: Software cams, position switching cycles - only 840D sl ... 693

9.1 Brief description ... 693

9.2 Cam signals and cam positions ... 694
9.2.1 Generation of cam signals for separate output ... 694
9.2.2 Generation of cam signals with gated output ... 698
9.2.3 Cam positions ... 702
9.2.4 Lead/delay times (dynamic cam) ... 704

9.3 Output of cam signals ... 706
9.3.1 Activating ... 706
9.3.2 Output of cam signals to PLC ... 706
9.3.3 Output of cam signals to NCK I/Os in position control cycle ... 707
9.3.4 Timer-controlled cam signal output ... 708
9.3.5 Independent, timer-controlled output of cam signals ... 710

9.4 Position-time cams ... 712

9.5 Supplementary conditions ... 714

9.6 Data lists ... 715
9.6.1 Machine data ... 715
9.6.1.1 General machine data ... 715
9.6.2 Setting data ... 715
9.6.2.1 General setting data ... 715
9.6.3 Signals ... 716
9.6.3.1 Signals to axis/spindle ... 716
9.6.3.2 Signals from axis/spindle ... 716

10 N4: Punching and nibbling - only 840D sl ... 717

10.1 Brief description ... 717

10.2 Stroke control ... 717
10.2.1 General information ... 717
10.2.2 High-speed signals ... 718
10.2.3 Criteria for stroke initiation ... 719
10.2.4 Axis start after punching ... 722

10.2.5 PLC signals specific to punching and nibbling....723
10.2.6 Punching and nibbling-specific reactions to standard PLC signals....723
10.2.7 Signal monitoring....724

10.3 Activation and deactivation....725
10.3.1 Language commands....725
10.3.2 Functional expansions....729
10.3.3 Compatibility with earlier systems....733

10.4 Automatic path segmentation....735
10.4.1 General information....735
10.4.2 Operating characteristics with path axes....736
10.4.3 Response in connection with single axes....741

10.5 Rotatable tool....746
10.5.1 General information....746
10.5.2 Coupled motion of punch and die....747
10.5.3 Tangential control....748

10.6 Protection zones....753

10.7 Supplementary conditions....753

10.8 Examples....754
10.8.1 Examples of defined start of nibbling operation....754

10.9 Data lists....760
10.9.1 Machine data....760
10.9.1.1 General machine data....760
10.9.1.2 Channel-specific machine data....760
10.9.2 Setting data....760
10.9.2.1 Channel-specific setting data....760
10.9.3 Signals....761
10.9.3.1 Signals to channel....761
10.9.3.2 Signals from channel....761
10.9.4 Language commands....761

11 P2: Positioning axes....763

11.1 Brief description....763

11.2 Own channel, positioning axis or concurrent positioning axis....766
11.2.1 Own channel - only 840D sl....766
11.2.2 Positioning axis (posAxis)....767
11.2.3 Concurrent positioning axis....769

11.3 Motion behavior and interpolation functions....771
11.3.1 Path interpolator and axis interpolator....771
11.3.2 Interpolation response of path axis in G0....771
11.3.3 Autonomous single-axis operations....773
11.3.4 Autonomous single-axis functions with NC-controlled ESR....781

11.4 Velocity....783

11.5 Programming....784
11.5.1 General....784
11.5.2 Revolutional feed rate in external programming....787

11.6 Block change....788
11.6.1 Settable block change time....791
11.6.2 End of motion criterion with block search....798

11.7 Control by the PLC ... 799
11.7.1 Starting concurrent positioning axes from the PLC ... 801
11.7.2 PLC-controlled axes ... 801
11.7.3 Control response of PLC-controlled axes ... 803

11.8 Response with special functions ... 804
11.8.1 Dry run (DRY RUN) ... 804
11.8.2 Single block ... 804

11.9 Examples ... 805
11.9.1 Motion behavior and interpolation functions ... 805
11.9.1.1 Traversing path axes without interpolation with G0 ... 806

11.10 Data lists ... 807
11.10.1 Machine data ... 807
11.10.1.1 Channel-specific machine data ... 807
11.10.1.2 Axis/spindle-specific machine data ... 807
11.10.2 Setting data ... 807
11.10.2.1 Axis/spindle-specific setting data ... 807
11.10.3 Signals ... 807
11.10.3.1 Signals to channel ... 807
11.10.3.2 Signals from channel ... 808
11.10.3.3 Signals to axis/spindle ... 808
11.10.3.4 Signals from axis/spindle ... 808

12 P5: Oscillation - only 840D sl ... 809

12.1 Brief description ... 809

12.2 Asynchronous oscillation ... 811
12.2.1 Influences on asynchronous oscillation ... 812
12.2.2 Asynchronous oscillation under PLC control ... 818
12.2.3 Special reactions during asynchronous oscillation ... 819

12.3 Oscillation controlled by synchronized actions ... 822
12.3.1 Infeed at reversal point 1 or 2 ... 825
12.3.2 Infeed in reversal point range ... 826
12.3.3 Infeed at both reversal points ... 827
12.3.4 Stop oscillation movement at the reversal point ... 828
12.3.5 Oscillation movement restarting ... 829
12.3.6 Do not start partial infeed too early ... 830
12.3.7 Assignment of oscillation and infeed axes OSCILL ... 831
12.3.8 Definition of infeeds POSP ... 831
12.3.9 External oscillation reversal ... 832

12.4 Marginal conditions ... 834

12.5 Examples ... 834
12.5.1 Example of asynchronous oscillation ... 834
12.5.2 Example 1 of oscillation with synchronized actions ... 835
12.5.3 Example 2 of oscillation with synchronized actions ... 838
12.5.4 Examples for starting position ... 840
12.5.4.1 Define starting position via language command ... 840
12.5.4.2 Start oscillation via setting data ... 840
12.5.4.3 Non-modal oscillation (starting position = reversal point 1) ... 841
12.5.5 Example of external oscillation reversal ... 843
12.5.5.1 Change reversal position via synchronized action with "external oscillation reversal" ... 843

12.6 Data lists ... 844
12.6.1 Machine data ... 844

12.6.1.1 General machine data....844
12.6.2 Setting data....844
12.6.2.1 Axis/spindle-specific setting data....844
12.6.3 Signals....844
12.6.3.1 Signals to axis/spindle....844
12.6.3.2 Signals from axis/spindle....845
12.6.4 System variables....845
12.6.4.1 Main run variables for motion-synchronous actions....845

13 R2: Rotary axes....849

13.1 Brief description....849
13.2 Modulo 360 degrees....854
13.3 Programming rotary axes....857
13.3.1 General information....857
13.3.2 Rotary axis with active modulo conversion (continuously-turning rotary axis).....857
13.3.3 Rotary axis without modulo conversion....863
13.3.4 Other programming features relating to rotary axes....865
13.4 Activating rotary axes....866
13.5 Special features of rotary axes....868
13.6 Examples....869
13.7 Data lists....870
13.7.1 Machine data....870
13.7.1.1 General machine data....870
13.7.1.2 Axis/spindle-specific machine data....870
13.7.2 Setting data....870
13.7.2.1 General setting data....870
13.7.2.2 Axis/spindle-specific setting data....870
13.7.3 Signals....871
13.7.3.1 Signals to axis/spindle....871
13.7.3.2 Signals from axis/spindle....871

14 S3: Synchronous spindle....873

14.1 Brief description....873
14.1.1 Function....873
14.1.2 Requirements....875
14.1.3 Synchronous mode....875
14.1.4 Prerequisites for synchronous mode....881
14.1.5 Selecting the parts program's synchronous mode....882
14.1.6 Deselecting the parts program's synchronous mode....885
14.1.7 Controlling synchronous spindle coupling via PLC....886
14.1.8 Monitoring of synchronous operation....889
14.2 Programming of synchronous spindle couplings....892
14.2.1 Preparatory programming instructions....892
14.2.2 Programming instructions for activating and deactivating the coupling....895
14.2.3 Axial system variables for synchronous spindle....897
14.2.4 Automatic selection and deselection of position control....898
14.3 Configuration of a synchronous spindle pair via machine data....900
14.3.1 Configuration of the behavior with NC start....901
14.3.2 Configuration of the behavior with Reset....902
14.4 Special features of synchronous mode....903

14.4.1 Special features of synchronous mode in general ... 903
14.4.2 Restore synchronism of following spindle ... 905
14.4.3 Influence on synchronous operation via PLC interface ... 907
14.4.4 Differential speed between leading and following spindles ... 910
14.4.5 Behavior of synchronism signals during synchronism correction ... 915
14.4.6 Delete synchronism correction and NC reset ... 915
14.4.7 Special points regarding start-up of a synchronous spindle coupling ... 916
14.5 Examples ... 921
14.6 Data lists ... 922
14.6.1 Machine data ... 922
14.6.1.1 NC-specific machine data ... 922
14.6.1.2 Channel-specific machine data ... 922
14.6.1.3 Axis/spindle-specific machine data ... 922
14.6.2 Setting data ... 923
14.6.2.1 Channel-specific setting data ... 923
14.6.3 Signals ... 923
14.6.3.1 Signals to channel ... 923
14.6.3.2 Signals from channel ... 923
14.6.3.3 Signals to axis/spindle ... 923
14.6.3.4 Signals from axis/spindle ... 924
14.6.4 System variables ... 924
14.6.4.1 System variables ... 924
15 S7: Memory configuration ... 925
15.1 Brief description ... 925
15.2 Memory organization ... 925
15.2.1 Active and passive file system ... 925
15.2.2 Reconfiguration ... 927
15.3 Configuration of the static user memory ... 928
15.3.1 Division of the static NC memory ... 928
15.3.2 Startup ... 930
15.4 Configuration of the dynamic user memory ... 932
15.4.1 Division of the dynamic NC memory ... 932
15.4.2 Startup ... 933
15.5 Data lists ... 934
15.5.1 Machine data ... 934
15.5.1.1 General machine data ... 934
15.5.1.2 Channel-specific machine data ... 937
15.5.1.3 Axis/spindle-specific machine data ... 939
16 T1: Indexing axes ... 941
16.1 Brief description ... 941
16.2 Traversing of indexing axes ... 942
16.2.1 General information ... 942
16.2.2 Traversing of indexing axes in the JOG mode ... 942
16.2.3 Traversing of indexing axes in the AUTOMATIC mode ... 944
16.2.4 Traversing of indexing axes by PLC ... 945
16.3 Parameterization of indexing axes ... 946
16.4 Programming of indexing axes ... 948
16.5 Equidistant index intervals ... 953

16.5.1 Function 953
16.5.2 Hirth tooth system 955
16.5.3 Response of the Hirth axes in particular situations 956
16.5.4 Restrictions 957
16.5.5 Modified activation of machine data 958
16.6 Starting up indexing axes 959
16.7 Special features of indexing axes 962
16.8 Examples 963
16.8.1 Examples of equidistant indexes 963
16.9 Data lists 965
16.9.1 Machine data 965
16.9.1.1 General machine data 965
16.9.1.2 Axis/spindle-specific machine data 965
16.9.2 Setting data 965
16.9.2.1 General setting data 965
16.9.3 Signals 966
16.9.3.1 Signals from axis/spindle 966
16.9.4 System variables 966
16.9.4.1 System variables 966
17 W3: Tool change 967
17.1 Brief description 967
17.2 Tool magazines and tool change equipments 968
17.3 Tool change times 968
17.4 Cut-to-cut time 968
17.5 Starting the tool change 969
17.6 Tool change point 970
17.7 Supplementary conditions 970
17.8 Examples 971
17.9 Data lists 973
17.9.1 Machine data 973
17.9.1.1 General machine data 973
17.9.1.2 Channel-specific machine data 973
17.9.1.3 Axis-/spindle-specific machine data 973
17.9.2 Signals 973
17.9.2.1 Signals from channel 973
18 W4: Grinding-specific tool offset and monitoring functions - only 840D sl 975
18.1 Brief description 975
18.2 Tool offset for grinding operations 976
18.2.1 Structure of tool data 976
18.2.2 Cutting-edge-specific offset data 978
18.2.3 Tool-specific grinding data 981
18.2.4 Examples of grinding tools 986
18.3 Online tool offset 991
18.3.1 General information 991
18.3.2 Write online tool offset: Continuous 993
18.3.3 Activate/deactivate online tool offset 996

18.3.4 Example of writing online tool offset continuously ... 997
18.3.5 Write online tool offset discretely ... 998
18.3.6 Information about online offsets ... 999

18.4 Online tool radius compensation ... 1001

18.5 Grinding-specific tool monitoring ... 1002
18.5.1 General information ... 1002
18.5.2 Geometry monitoring ... 1003
18.5.3 Speed monitoring ... 1004
18.5.4 Selection/deselection of tool monitoring ... 1005

18.6 Constant grinding wheel peripheral speed (GWPS) ... 1006
18.6.1 General information ... 1006
18.6.2 Selection/deselection and programming of GWPS, system variable ... 1007
18.6.3 GWPS in all operating modes ... 1008
18.6.4 Example of how to program GWPS ... 1009

18.7 Supplementary conditions ... 1011
18.7.1 Tool changes with online tool offset ... 1011

18.8 Data lists ... 1011
18.8.1 Machine data ... 1011
18.8.1.1 General machine data ... 1011
18.8.1.2 Channel-specific machine data ... 1011
18.8.1.3 Axis/spindle-specific machine data ... 1011
18.8.2 Signals ... 1012
18.8.2.1 Signals from axis/spindle ... 1012

19 Z2: NC/PLC interface signals ... 1013

19.1 Digital and analog NCK I/Os ... 1013
19.1.1 Signals to NC (DB10) ... 1013
19.1.2 Signals from NC (DB10) ... 1021

19.2 Several operator panels on several NCUs, distributed systems ... 1025
19.2.1 Defined logical functions/defines ... 1025
19.2.2 Interfaces in DB19 for M:N ... 1028
19.2.3 Signals from NC (DB10) ... 1034
19.2.4 Signals from axis/spindle (DB31, ...) ... 1035

19.3 Operation via PG/PC (B4) ... 1035

19.4 Manual and handwheel travel ... 1036
19.4.1 Signals from NC (DB10) ... 1036
19.4.2 Signals to channel (DB21, ...) ... 1039
19.4.3 Signals from channel (DB21, ...) ... 1044
19.4.4 Signals with contour handwheel ... 1050
19.4.5 Signals to axis/spindle (DB31, ...) ... 1052
19.4.6 Signals from axis/spindle (DB31, ...) ... 1057

19.5 Compensations (K3) ... 1061

19.6 Mode groups, channels, axis replacement ... 1062
19.6.1 Signals to axis/spindle (DB31, ...) ... 1062
19.6.2 Signals from axis/spindle (DB31, ...) ... 1062

19.7 Kinematic transformation ... 1063
19.7.1 Signals from channel (DB21, ...) ... 1063

19.8 Measurement ... 1064
19.8.1 Signals from NC (DB10) ... 1064

19.8.2 Signals from axis/spindle (DB31, ...) 1064

19.9 Software cams, position switching signals 1065
19.9.1 Signal overview 1065
19.9.2 Signals from NC (DB10) 1066
19.9.3 Signals to axis/spindle (DB31, ...) 1067
19.9.4 Signals from axis/spindle (DB31, ...) 1067

19.10 Punching and nibbling 1068
19.10.1 Signal overview 1068
19.10.2 Signals to channel (DB21, ...) 1068
19.10.3 Signals from channel (DB21, ...) 1070

19.11 Positioning axes 1072
19.11.1 Signals to axis/spindle (DB31, ...) 1072
19.11.2 Function call - only 840D sl 1075

19.12 Oscillation 1076
19.12.1 Signals to axis/spindle (DB31, ...) 1076
19.12.2 Signals from axis/spindle (DB31, ...) 1077

19.13 Rotary axes 1079
19.13.1 Signals to axis/spindle (DB31, ...) 1079
19.13.2 Signals from axis/spindle (DB31, ...) 1079

19.14 Synchronous spindle 1080
19.14.1 Signals to axis/spindle (DB31, ...) 1080
19.14.2 Signals from axis/spindle (DB31, ...) 1080

19.15 Memory configuration (S7) 1083

19.16 Indexing axes 1083
19.16.1 Signals from axis/spindle (DB31, ...) 1083

19.17 Tool change (W3) 1084

19.18 Grinding-specific tool offset and tool monitoring 1084
19.18.1 Signals from axis/spindle (DB31, ...) 1084

A Appendix 1087

A.1 List of abbreviations 1087

A.2 Feedback on the documentation 1095

A.3 Overview 1097

Glossary 1099

Index 1121

A4: Digital and analog NCK I/Os

1

1.1 Brief description

General information

Signals can be read and output in the interpolation cycle via the "digital and analog NCK I/Os". The following functions can be executed with these signals, for example:

- Several feedrate values in one block
- Several auxiliary functions in one block
- Rapid retraction on final dimension
- Axis-specific delete distance-to-go
- Program branches
- Rapid NC Start
- Analog calipers
- Position switching signals
- Punching/nibbling functions
- Analog-value control

Access to NCK I/Os:

- NCK I/O via PLC: for Onboard/and PROFIBUS-I/Os
- NCK I/O via PROFIBUS: for PROFIBUS I/Os

Advantages of the function "NCK-I/Os via PROFIBUS"

With reference to the function "NCK-I/Os via PLC" the following are the advantages for the direct communication of the NCK with the PROFIBUS-I/Os:

- Direct access to the PROFIBUS-I/O, without diversion via the PLC.
- The PLC-operating system is not being loaded.
- Reading is possible simultaneously from part program/synchronous actions and complie cycles.
- The configured I/O areas must not follow one another contiguously.

1.2 NCK I/O via PLC

1.2.1 General functionality

General

The ability to control or influence time-critical NC functions is dependent on high-speed NCK I/O interfaces or the facility to rapidly address particular PLC I/Os.

Therefore, the functions below can be carried out using the SINUMERIK 840D/840Di:

- Use of digital and analog NCK inputs/outputs
- Direct addressing of particular PLC I/Os

The hardware inputs/outputs can be read and written via system variables in the part program or synchronized actions.

Via the PLC interface, both the signal states of the digital I/Os and the values of the external analog I/Os can be changed by the PLC user program according to the application.

840D hardware

On the SINUMERIK 840D **on-board NCU** there are four digital NCK inputs (inputs 1 to 4) and four digital NCK outputs (outputs 1 to 4).

These four digital on-board inputs/outputs are stored in the first address byte. With the NCK outputs, the remaining signals of this byte (NCK outputs 5 to 8) can be used via the PLC interface (digital NCK outputs without hardware).

It is possible to connect further digital and analog NCK inputs/outputs (hereafter called **external NCK I/Os**) using the "NCU terminal block", which can be coupled to the drive bus.

The "NCU terminal block" is used as a carrier module for up to eight DP compact plug-in modules. Up to two "NCU terminal blocks" can be connected per NCU.

The maximum degree of expansion of the external NCK I/Os is:

- 32 digital NCK inputs (digital inputs 9 to 40)
- 32 digital NCK outputs (digital outputs 9 to 40)
- Eight analog NCK inputs (analog inputs 1 to 8)
- Eight analog NCK outputs (analog outputs 1 to 8)

For further information about the hardware specification see:

References:

/PHD/ SINUMERIK 840D Configuration Manual (HW)

840Di hardware digital I/Os

Digital inputs/outputs are provided for the SINUMERIK 840Di via the MCI-Board-Extension module. The following connections are available:

- Two handwheels
- Two probes
- Four digital inputs/outputs

Note

The MCI-Board-Extension module is an option for the SINUMERIK 840Di. The pin assignment of the cable-distributor interface (X121) matches that of the cable distributor on the SINUMERIK 840D.

SINUMERIK 840Di analog and digital inputs/outputs

Analog and digital inputs/outputs can be operated on the SINUMERIK 840Di by means of SIMATIC S7 bus-interface and signal modules, via the PROFIBUS-DP.

PLC I/Os for direct addressing by NCK

Up to 16 bytes for digital input signals and analog input values plus a total of 16 bytes for digital output signals and analog output values can be addressed directly by the part program. These bytes must be taken into account when the PLC is configured. They must be programmed consecutively. They are processed directly by the PLC operating system. As a result, the time taken to transfer signals between the NC and PLC I/O modules is of an order of magnitude of 0.5 ms.

Note

The output bytes specified for the NCK may not be write-accessed by the PLC user program, as the access operations between the NCK and PLC would be uncoordinated.

Comparator inputs

In addition to the digital and analog NCK inputs, 16 internal comparator inputs (comparator input bytes 1 and 2) are also available.

The signal state of a comparator input is formed by comparing an analog input signal with a threshold value within a setting data.

Number

The number of addressable digital NCK input/output bytes and analog inputs/outputs must be defined by means of general machine data.

Machine data	Number of active ...	Max. number
MD10350 $MN_FASTIO_DIG_NUM_INPUTS	... Digital NCK input **bytes**	5
MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS	... Digital NCK output **bytes**	5
MD10300 $MN_FASTIO_ANA_NUM_INPUTS	... Analog NCK inputs	8
MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS	... Analog NCK outputs	8

Note

On the SINUMERIK 840Di, the first byte is always assigned to the four digital I/Os on the MCI-Board-Extension module. Even if you do not have an MCI-Board-Extension module connected to the SINUMERIK 840Di, the first byte will still be occupied by this module.

For this reason an entry of at least two bytes must be made into the machine data:

$MN_FASTIO_DIG_NUM...

if you want to operate additional I/O peripherals via the PROFIBUS.

Corresponding alarms are generated if the part program addresses inputs/outputs that have not been defined in the above machine data.

These NCK inputs/outputs do not have to actually exist in the hardware.

In this case the signal states or the binary analog values are set to "zero" in a defined way inside the NCK. However, these values can be changed by the PLC.

Hardware assignment of the external NCK I/Os

The following general machine data are provided for assigning I/O signal modules or I/O modules to external NCK I/Os:

MD10366 $MN_HW_ASSIGN_DIG_FASTIN[hw]

(H/W assignment for external digital inputs)

MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[hw]

(H/W assignment for external digital outputs)

MD10362 $MN_HW_ASSIGN_ANA_FASTIN[hw]

(H/W assignment for external analog inputs)

MD10364 $MN_HW_ASSIGN_ANA_FASTOUT[hw]

(H/W assignment for external analog outputs)

[hw]: Index for addressing the external digital I/O bytes (0 to 3) or the external analog inputs/outputs (0 to 7)

Note

The hardware assignment is different on the SINUMERIK 840D and 840Di controls.

The **assignment** of I/Os for SINUMERIK 840Di is specified via the machine data:

MD10362 to MD10368

with the following default values:

Machine data	Description	Default
MD10362 $MN_HW_ASSIGN_ANA_FASTIN[0] ...	Assignment of analog input (16-bit access)	050000A0
MD10364 $MN_HW_ASSIGN_ANA_FASTOUT[0] ...	Assignment of analog output (16-bit access)	050000A0
MD10366 $MN_HW_ASSIGN_DIG_FASTIN[0] ...	Assignment of digital input (8-bit access)	05000090
MD10368 $MN_HW_ASSIGN_DIG_FASTOUT[0] ...	Assignment of digital output (8-bit access)	05000090

Modification to MD for PROFIBUS-DP

The following machine data has been modified for hardware operation on the PROFIBUS-DP of the SINUMERIK 840Di.

$MN_HW_ASSIGN_...

The assignment of bytes 1 to 4 has been redefined. The machine data assignments below apply for PROFIBUS-DP operation:

Byte	New for PROFIBUS-DP	Previous meaning
4 bytes	Segment number = 5	Segment number
3 bytes	Not used = 0	Module number
2 bytes	Logical address high	Interface-module number
1 byte	Logical address low	Input/output number

Guidelines for machine data $MN_HW_ASSIGN_...:

- Logical address in 1st and 2nd byte is specified in hexadecimal format.

  Example: 050001A2 (hex) equals logical address 418 (dec).
- Address 0 is reserved for the PLC and cannot be used as an NC I/O.
- The value 05000000 in the machine data:
- $MN_HW_ASSIGN_...
- represents the signal "Slot is not present physically".
- The input is then treated like a simulation input.

System variable

The following table lists the system variables with which NCK I/Os can be read or written directly by the part program.

The number of the NCK input/output is used for addressing.

Applies to n:

1 ≤ n ≤ 8 * MD10350 $MN_FASTIO_DIG_NUM_INPUTS

1 ≤ n ≤ 8 * MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

1 ≤ n ≤ MD10300 $MN_FASTIO_ANA_NUM_INPUTS

1 ≤ n ≤ MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

System variable	Description	Range of [n]
$A_IN[n]	Read digital NCK input [n]	1 to 3, 9 to 40
$A_INA[n]	Read analog NCK input [n]	1 to 8
$A_INCO[n]	Read comparator input [n]	1 to 16
PBB		
$A_OUT[n]	Read/write digital NCK output [n]	1 to 40
$A_OUTA[n]	Read/write analog NCK output [n]	1 to 8

Note

When these system variables are read by the part program, a preprocessing stop (`STOPRE` command) is initiated inside the control.

Weighting factor

The evaluation factors of the following general machine data allow each individual analog NCK input and output to be adapted to the AD or DA converters of the analog I/O module used:

MD10320 $MN_FASTIO_ANA_INPUT_WEIGHT[hw]

MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[hw]

If the correct weighting factor is set, the value set in system variable $A_OUTA[n] generates the corresponding voltage value in millivolts at analog output [n].

Example for 840D

Analog-value range is 10 V (maximum modulation);

MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[hw] = 10000

(standard value for 840D)

$A_OUTA[1] = 9500 ; 9.5 V is output at analog NCK output 1

$A_OUTA[3] = -4120 ; -4.12 V is output at analog NCK output 3

Application for analog NCK inputs/outputs without hardware:

With a weighting factor of 32767, the digitized analog values for parts program and PLC access are identical. In this way, it is possible to use the associated input or output word for 1:1 communication between the part program and the PLC.

Assignment to NC functions

Several NC functions are dependent on the functionality of the NCK I/Os.

Assignment of NCK inputs and outputs for these NC functions

is accomplished function-specifically via machine data, for instance:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN (Multiple feedrates in one batch).

A byte address should be specified in the machine data for digital inputs/outputs. Each byte is assigned separately.

Byte address	Assignment for the digital NCK inputs/outputs			
0	None			
1	1 to 4 (on-board I/O)	and	5 to 8	(NCK output without hardware)
2	9	to	16	(external NCK I/Os)
3	17	to	24	(external NCK I/Os)
4	25	to	32	(external NCK I/Os)
5	33	to	40	(external NCK I/Os)
128	Inputs 1 to 8 of comparator byte 1			
129	Inputs 9 to 16 of comparator byte 2			

Isochronous processing

The I/O modules of the external NCK I/Os on the SINUMERIK 840D can be operated in one of the following two modes:

- **Asynchronous**

  The input and output values are made available in cycles set by the terminal block, which are asynchronous with the internal NC processing cycles.

- **Synchronous**

  The input and output values are made available in synchronism with a settable internal NC processing cycle.

The processing mode is selected for individual modules by means of general machine data:

MD10384 $MN_HW_CLOCKED_MODULE_MASK[tb]

[tb] = Index for terminal block (0 to 1)

In synchronous processing mode, one of the following clock rates can be selected

(MD10380 $MN_HW_UPDATE_RATE_FASTIO[tb]):

- Synchronous inputs/outputs in position-control cycle (default setting)
- Synchronous inputs/outputs in interpolation cycle

It is possible to define a lead time in microseconds for the clocked NCK I/Os in general machine data:

MD10382 $MN_HW_LEAD_TIME_FASTIO[tb].

This makes it possible to consider the conversion time of the analog-to-digital converter, for example, so that the digitized input value is available at the cycle time.

The defined clock rate or rate time applies to all isochronous I/O modules of the terminal block addressed with [tb].

Monitoring

The following functional monitors are provided for external I/Os on the SINUMERIK 840D:

- During booting:
  - Check whether the arrangement of components of the I/O modules in the terminal blocks matches the MD assignments.
- During cyclic operation:
  - Sign-of-life monitoring in interpolation cycles
  - Module monitoring in interpolation cycles
  - Temperature monitoring

In the event of a fault, NC-Ready is canceled and an alarm is output.

Response to faults

In case of malfunction the digital and analog NCK outputs

are switched to "safe" status (0 V at output), in the event of faults (e.g., NC-Ready = 0) in the NCU or power failures.

1.2.2 NCK digital inputs/outputs

1.2.2.1 NCK digital inputs

Number

General machine data is used to define available digital NCK inputs (in groups of 8).

MD10350 $MN_FASTIO_DIG_NUM_INPUTS

(Number of active digital NCK input bytes)

Function

The digital NCK inputs allow external signals to be injected which can then be used, for example, to control the workpiece-machining program sequence.

The signal state of digital input [n] can be scanned directly in the part program using system variable **$A_IN[n]**.

The signal state at the hardware input can be changed by the PLC user program (see fig.).

Disable input

The digital NCK inputs can be disabled individually from the PLC application program with the following
interface signals:

DB10 DBB0 or DBB122 ... (Disabling of digital NCK inputs)

In this case, they are set to "0" in a defined manner inside the control.

Set input from PLC

The PLC can also use the following interface signal to set each digital input to a defined "1" signal (see figure):
DB10 DBB1 or DBB123 ... (PLC setting of digital NCK inputs)

As soon as this interface signal is set to "1", the signal state at the hardware input or the input disable is inactive.

Read actual value

The signal state of the digital NCK inputs is sent to the PLC:

DB10, DBB60 or DBB186 ... (actual value for digital NCK inputs)

The actual value reflects the actual state of the signal at the hardware input. The influence of the PLC is, therefore, ignored in the actual value (see fig.).

RESET/POWER ON response

After POWER ON and RESET, the signal level at the respective input is passed on. If necessary, the PLC user program can disable or set the individual inputs to "1" in a defined manner as described above.

Applications

The program sequence can be controlled with conditional jump statements in the part program as a function of an external-hardware-signal state.

For example, digital NCK inputs can be used for the following NC functions:

- Delete distance-to-go with positioning axes
- Fast program branching at the end of block
- Programmed read-in disable
- Several feedrates in one block

References:
/FBSY/ Function Manual; Synchronized Actions

The NCK inputs are assigned to the NC functions separately for each function and byte via machine data. Multiple assignments of inputs are not monitored.

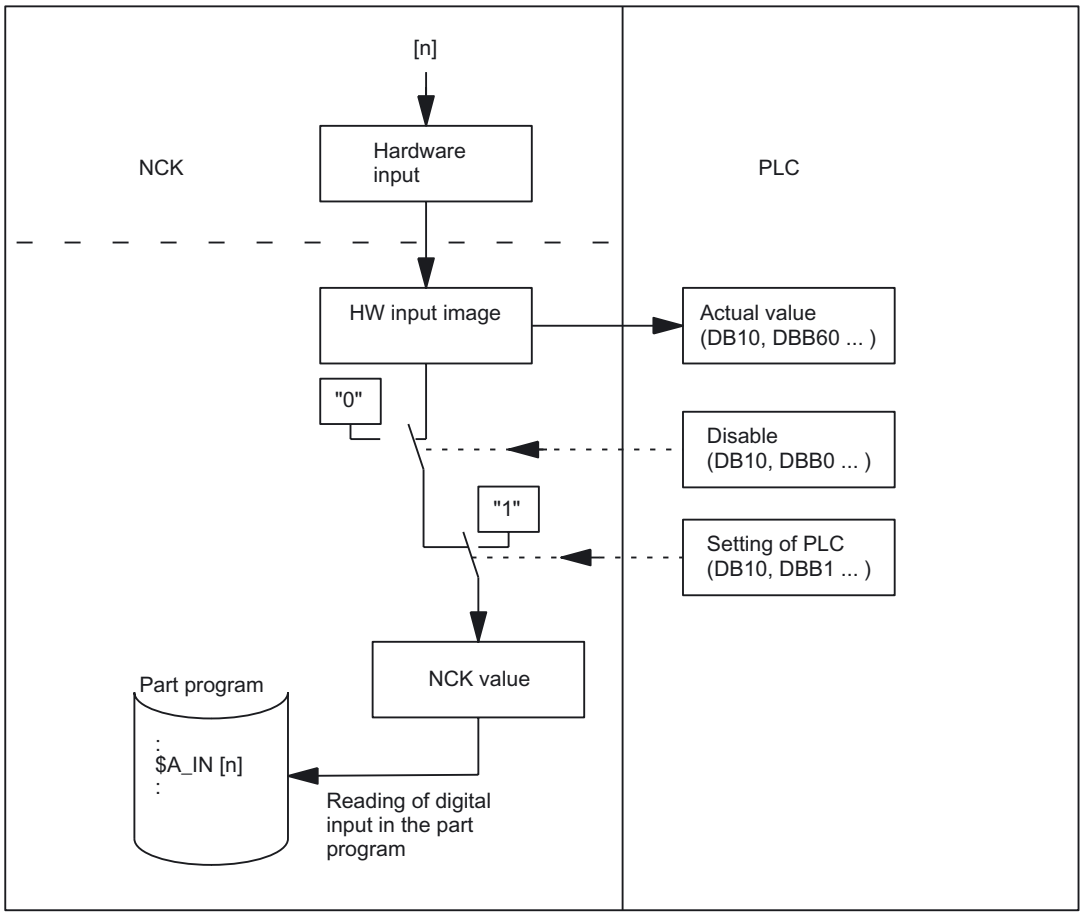


Figure 1-1 Signal flow for digital NCK inputs

1.2.2.2 NCK digital outputs

Number

The available digital NCK outputs can be defined (in groups of eight) using the following general machine data (number of active digital NCK output bytes):

MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

Function

The digital NCK outputs provide the option of outputting important switching commands at high speed as a function of the program processing status.

The signal state of digital output [n] can be set or read again directly in the part program using system variable **$A_OUT[n]**.

There are also several ways of changing this set signal state via the PLC (see fig.).

Disable output

The PLC user program is capable of disabling the digital NCK outputs individually with interface signal
DB10, DBB4 or DBB130 ... (disable digital NCK outputs).
In this case, the "0" signal is output at the hardware output (see fig.).

Overwrite mask

Every output that can be set by the NC part program can be **overwritten** from the PLC using the overwrite mask. The previous NCK value is then lost (see fig.).

The following routine has to be carried out to overwrite the NCK value from the PLC:

1. The relevant PLC interface output has to be set to the required signal status:
   DB10 DBB6 or DBB132 ... (Setting value by PLC of digital NCK outputs)

2. The setting value becomes the new NCK value when the overwrite mask for the relevant output (DB10, DBB5, or DBB131 ...) is activated (signal transition 0 → 1). This value remains operative until a new NCK value is programmed (by the PLC or from the NC part program).

Setting mask

Furthermore, a PLC setting for each output can determine whether the current NCK value (e.g., as specified by the NC part program) or the PLC value specified via the setting mask (DB10, DBB7 or DBB133 ...) should be sent to the hardware output (see fig.).

The following routine has to be carried out to define the PLC value:

1. The output in question must be preset with the required signal state at the PLC interface DB10, DBB6 (PLC setting value for digital NCK outputs).
2. The setting mask must be set to "1" for the output in question.

Unlike the overwrite mask, the NCK value is not lost when a value is set in the setting mask. As soon as the PLC sets "0" in the setting mask, the NCK value becomes active again.

Note

The same setting value (DB10, DBB6) is used at the PLC interface for the overwrite and setting masks. Therefore, an identical output signal state is the result if the signal state is changed **simultaneously** in the overwrite **and** setting maska.

Read setpoint

The current NCK value of the digital outputs can be read by the PLC user program:

DB10, DBB64 or DBB186 ... (setpoint for digital NCK outputs)

Please note that this setpoint ignores disabling and the PLC setting mask. Therefore, the setpoint can differ from the actual signal state at the hardware output (see fig.).

RESET/end of program

At the end of the program or on RESET, a specific default value can be assigned by the PLC user program to every digital output in accordance with requirements, using the overwrite mask, setting mask or disable signal.

POWER ON

After POWER ON, the digital outputs are set to "0" in a defined manner. This can be overwritten from the PLC user program according to the specific application using the masks described above.

Digital NCK outputs without hardware

If digital NCK outputs, as defined in the following general machine data, are written by the part program, but are not available as hardware, no alarm is output. The NCK value can be read by the PLC (IS "setpoint ...").:

MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

Applications

This function allows digital hardware outputs to be set instantaneously by bypassing the PLC cycles. Time-critical switching functions can thus be triggered in connection with the machining process and under program control (e.g., on block change).

For example, digital NCK outputs are required for the following NC functions:

- Position-switching signals

  References:
  /FB2/ Function Manual, Extended Functions; Software Cam, Position-Switching Signals (N3)

- Output of comparator signals

The NCK outputs are assigned to the NC functions separately for each function via machine data. Multiple assignments of outputs are checked during booting and indicated by an alarm.

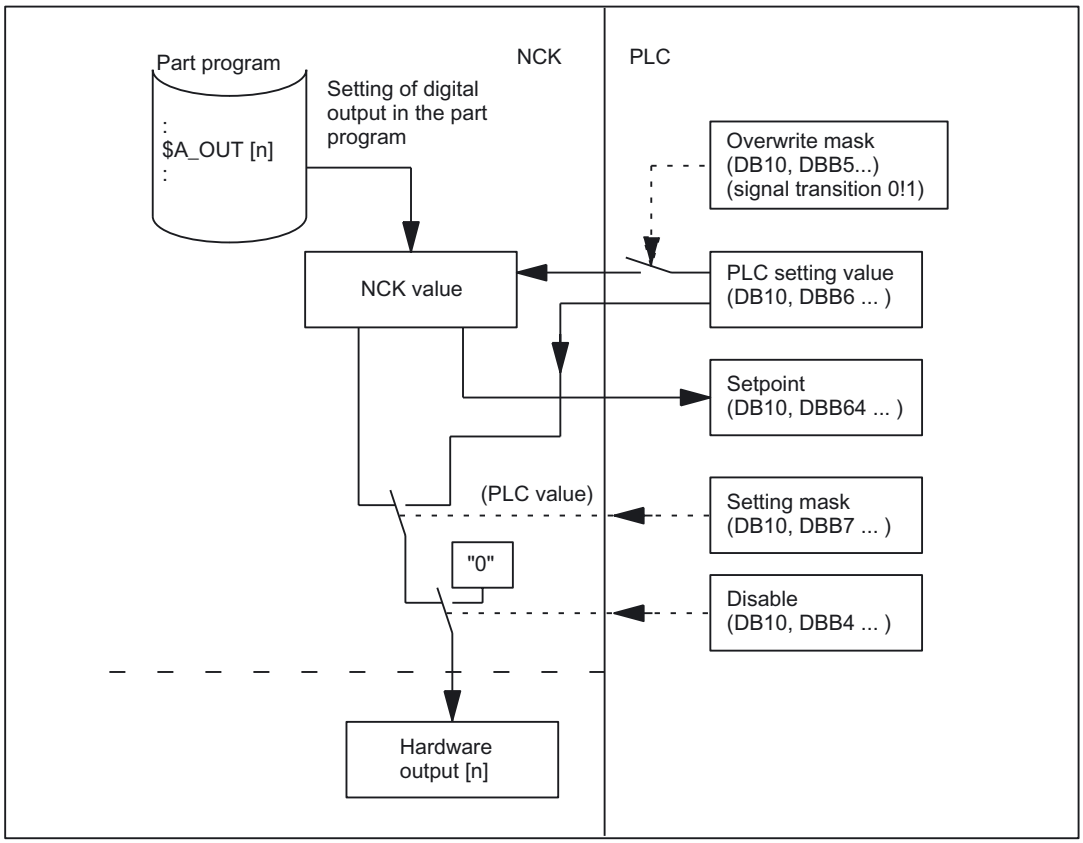


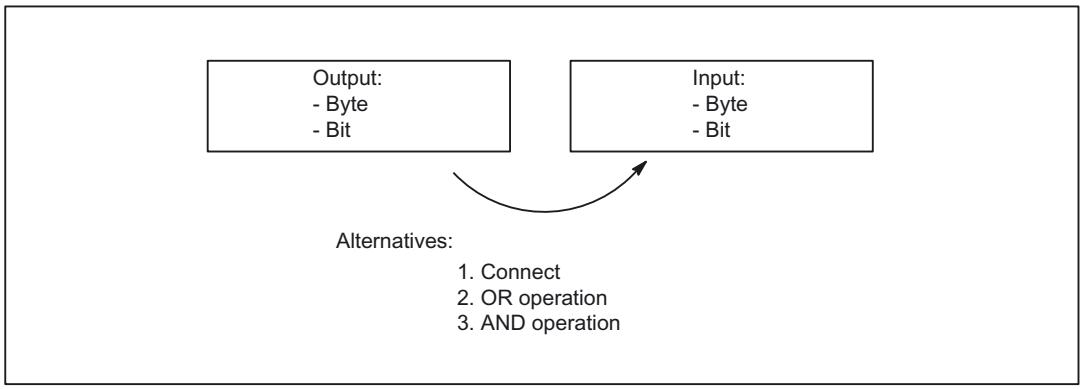
Figure 1-2 Signal flow for digital NCK outputs

1.2.3 Connection and logic operations of fast NCK inputs/outputs

Function

Fast NCK I/O inputs can be set using software as a function of fast-output signal states.

Overview:



Connect

The NCK I/O fast input is set to the signal state of the assigned fast output.

OR operation

The NCK I/O fast input adopts the signal state as a result of the ORing of the output signal with the assigned input signal.

AND operation

The NCK I/O fast input adopts the signal state as a result of the ANDing of the output signal with the assigned input signal.

Special cases

- If several output bits are assigned to the same input bit, then the one with the highest MD index becomes effective.
- If inputs or outputs are specified which do not exist or are not activated, then the assignment is ignored without an alarm. Checking of the active bytes of the NCK I/Os is performed via the entries in machine data:

  MD10350 $MN_FASTIO_DIG_NUM_INPUTS

  MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS.

Defining assignments

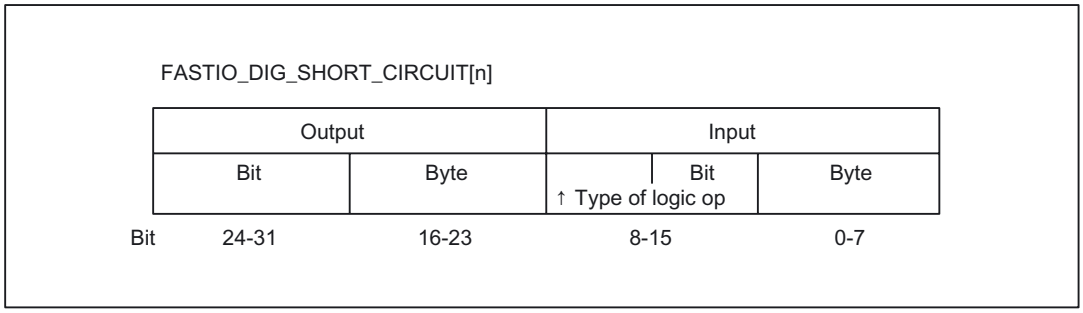
The assignments are specified via machine data:
MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT[n]

n: can accept values 0 to 9, so up to **10** assignments can be specified.

Two hexadecimal characters are provided for specifying the byte and bit of an output and an input.

Specifying 0, A and B in input bits 12 - 15 results in the following **logic operations**:

0	Connect
A	AND operation
B	OR operation



Examples

Connect:

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '04010302H'

Output 4, byte 1, connect to

Input 3, byte 2

AND operation:

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '0705A201H'

Output 7, byte 5 AND operation with

Input 2, byte 1

OR operation:

MD10361 $MN_FASTIO_DIG_SHORT_CIRCUIT = '0103B502H'

Output 1, byte 3 OR operation with

Input 5, byte 2

1.2.4 NCK analog inputs/outputs

1.2.4.1 NCK analog inputs

Amount

General machine data is used to define
available analog NCK inputs:

MD10300 $MN_FASTIO_ANA_NUM_INPUTS(Number of analog NCK inputs)

Function

The value of the analog NCK input [n] can be accessed directly in the part program using system variable **$A_INA[n]**.

The analog value at the hardware input can also be influenced by the PLC user program (see fig.).

Disable input

The PLC user program is capable of disabling the NCK inputs individually using interface signal
DB10 DBB146 (disable analog NCK inputs).
In this case, they are set to "0" in a defined manner inside the control.

Set input from PLC

The PLC can also specify a value for each analog NCK input using interface signal
DB10 DBB147 (setting mask for analog NCK inputs)
(see fig.).

As soon as this interface signal is set to "1", the setting value set by the PLC (DB10, DBB148 to 163) becomes active for the corresponding analog input. The analog value at the hardware input or the input disable is then inactive.

Read actual value

The analog values present at the hardware inputs are transferred to the PLC with the following interface signal:
DB10 DBB194-209 (Actual value of the analog NCK input).
The possible influence of the PLC thus has no effect on the actual value (see figure).

RESET/POWER ON response

After `POWER ON` and `RESET`, the analog value at the input is passed on. If necessary, the PLC user program can manipulate the individual analog NCK inputs via the PLC user program, as described above.

Weighting factor

The weighting factor in general machine data can be used to adapt the analog NCK inputs to different analog-to-digital converter hardware variants for the purpose of reading in the part program (see figure):

MD10320 $MN_FASTIO_ANA_INPUT_WEIGHT[hw]

In this machine data, it is necessary to enter the value x that is to be read in the part program with the system variable $A_INA[n], if the corresponding analog input [n] is set to the maximum value or if the value 32767 is set for this input via the PLC interface. The voltage level at the analog input is then read with system variable $A_INA[n] as a numerical value with the unit millivolts.

Binary analog-value display

See "Analog-value representation of the NCK analog input/output values".

Analog NCK input without hardware

The following values are read in the case of parts-program access to analog NCK inputs that are defined via machine data, but are not available as hardware inputs:

MD10300 $MN_FASTIO_ANA_NUM_INPUTS

- The setpoint set by the PLC,
  if the IS "PLC setting value for analog NCK inputs"
  is set to "1" (see fig.).
- Otherwise 0 volts

This makes it possible to use the functionality of the analog NCK inputs from the PLC user program without I/O hardware.

Applications

The analog NCK inputs are used particularly for grinding and laser machines (e.g., for the "analog calipers" NC function).

Fast analog NCK inputs

The fast analog inputs must be isochronous.
The assignment is defined by the machine data:

MD10384 $MN_HW_CLOCKED_MODULE_MASK

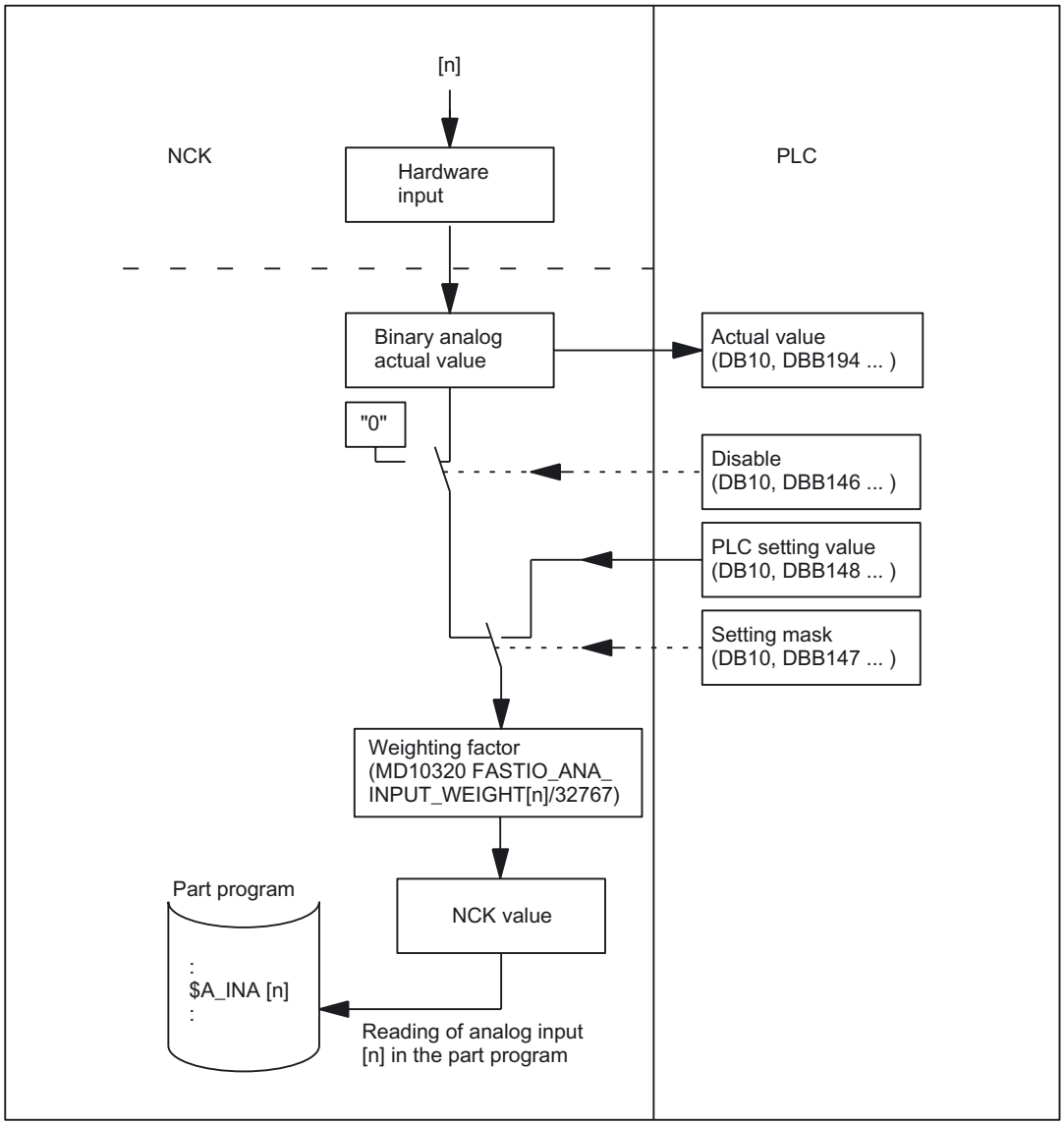


Figure 1-3 Signal flow for analog NCK inputs

1.2.4.2 NCK analog outputs

Number

The available analog NCK outputs are defined using general machine data MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS (number of analog NCK outputs).

Function

The value of the analog output [n] can be defined directly in the part program using system variable **$A_OUTA[n]**.

Before output to the hardware I/Os, the analog value set by the NCK can be changed by the PLC (see fig.).

Disable output

The PLC user program is capable of disabling the NCK outputs individually using interface signal
DB10 DBB168 (disable analog NCK outputs).
In this case, **0 volts** is output at the analog output (see fig.).

Overwrite mask

Every NCK analog value set by the NC part program can be **overwritten** from the PLC using the overwrite mask. The previous NCK value is then lost (see fig.).

The following routine has to be carried out to overwrite the NCK value from the PLC:

1. The output n in question must be preset with the required analog value at PLC interface DB10 DBB170 - 185 (PLC setting value for NCK analog output n).

2. The PLC setting value becomes the new NCK value when the overwrite mask for the relevant analog output (DB10, DBB166) is activated (signal transition 0 → 1).

This value remains valid until a new analog value is set for the NCK by the part program, for example.

Setting mask

Furthermore, a PLC setting for each output can determine whether the current NCK value (e.g., as specified by the NC part program) or the PLC value specified via the setting mask (DB10, DBB167) should be sent to the hardware analog output (see fig.).

The following routine has to be carried out to define the PLC value:

1. The output n in question must be preset with the required analog value at PLC interface DB10 DBB170 - 185 (PLC setting value for NCK analog output n).

2. The setting mask (DB10, DBB167) must be set to "1" for the analog output in question.

Unlike the overwrite mask, the current NCK value is not lost when a value is set in the setting mask. As soon as the PLC sets "0" in the corresponding setting mask, the NCK value becomes active again.

Note

The same setting value (DB10, DBB170 - 185) is used at the PLC interface for the overwrite and setting masks.

Read setpoint

The current NCK value of the analog outputs can be read by the PLC user program:

DB10, DBB210 - 225 (setpoint of NCK analog output n)

Please note that this setpoint ignores disabling and the PLC setting mask. Therefore, the setpoint can differ from the actual analog value at the hardware output (see fig.).

RESET/end of program

At the end of the program or on RESET, a specific default value can be assigned by the PLC user program to every analog output in accordance with requirements, using the overwrite mask, setting mask or disable signal.

POWER ON

After `POWER ON` the analog outputs are set to "0" in a defined manner. After booting, this can be overwritten in the PLC user program according to the application, using the masks described above.

Weighting factor

The weighting factor in general machine data
MD10330 $MN_FASTIO_ANA_OUTPUT_WEIGHT[hw].
can be used to adapt the analog NCK outputs to the different digital-to-analog converter hardware variants for the purpose of programming in the part program (see fig.).

In this machine data, it is necessary to enter the value x that is to cause the analog output [n] to be set to the maximum value or the value 32767 to be set for this output in the PLC interface, if $A_OUTA[n] = x is programmed. The value set with system variable $A_OUTA[n] then generates the corresponding voltage value at the analog output in millivolts.

Binary analog-value display

See "Analog-value representation of the NCK analog input/output values".

Special case

If values for NCK analog outputs defined in machine data
MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS
are programmed in the part program, but are not available as hardware, no alarm is output. The NCK value can be read by the PLC (IS "setpoint ...").

Application

This function allows analog values to be output instantaneously by bypassing the PLC cycles.

The analog NCK outputs are used in particular for grinding and laser machines.

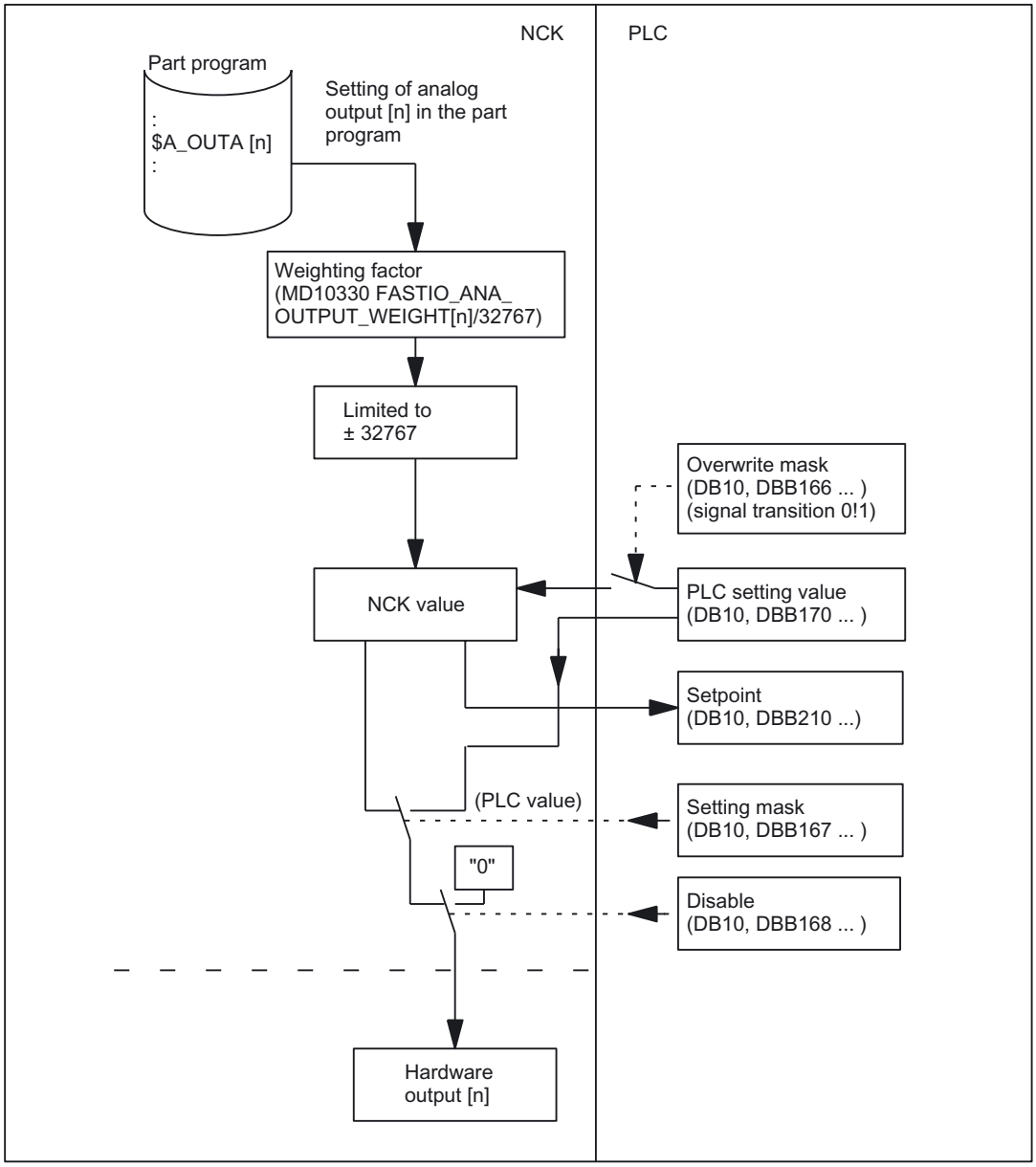


Figure 1-4 Signal flow for analog NCK outputs

1.2.5 Direct PLC I/Os, addressable from the NC

Introduction

The **fast data channel** between the NCK and PLC I/Os is processed **directly** and, therefore, quickly by the PLC operating system.

There is no provision for control of the PLC basic and user programs. It is not advisable for the NCK and the PLC to attempt to access the same PLC I/Os simultaneously, as this can result in faults.

System variables

For access purposes, the NC uses system variables associated with **part programs** and **synchronized actions**.

For **reading** from PLC:

$A_PBB_IN[n]	; Read byte (8-bit)
$A_PBW_IN[n]	; Read word (16-bit)
$A_PBD_IN[n]	; Read data word (32-bit)
$A_PBR_IN[n]	; Read real (32-bit float)
n	Byte offset within the PLC input area

Reading from the part program causes a preprocessing stop.

For **writing** to PLC:

$A_PBB_OUT[n]	; Write byte (8-bit)
$A_PBW_OUT[n]	; Write word (16-bit)
$A_PBD_OUT[n]	; Write data word (32-bit)
$A_PBR_OUT[n]	; Write real (32-bit float)
n	Byte offset within the PLC output area

The output data can also be read from the part program and synchronized actions. Reading from the part program causes an automatic preprocessing stop (to achieve synchronization with the real-time context).

Variable-value ranges

Values within the following ranges can be stored in the variables:

$A_PBB_OUT[n]	;(-128 ... +127) or (0 ... 255)
$A_PBW_OUT[n]	;(-32768 ... +32767) or (0 ... 65535)
$A_PBD_OUT[n]	;(-2147483648 ... +2147483647) or (0 ... 4294967295)
$A_PBR_OUT[n]	;(-3.402823466E+38 ... +3.402823466E+38)

Transfer times

Data is output from NCK ⇒ PLC (write) at the end of the interpolation cycle if at least one data was written.

Data is read in by transmitting a request at the end of the interpolation cycle, as a function of machine data.

MD10398 $MN_PLCIO_IN_UPDATE_TIME

The new data are available in the subsequent interpolation cycle at the earliest.

The time period within which a request is sent to the PLC can be set, using the following machine data.

MD10398 $MN_PLCIO_IN_UPDATE_TIME

The entered time period is rounded up internally to the next highest multiple of an interpolation cycle. If the value of these machine data is set to 0, the request will continue to be sent to the PLC in every interpolation cycle.

Configuring

To activate the functionality, the following machine data (Power ON active) must be configured on the NC:

MD10394 $MN_PLCIO_NUM_BYTES_IN

Number of PLC-I/O input bytes that are read directly by the NC

MD10395 $MN_PLCIO_LOGIC_ADDRESS_IN

Logical start address of the PLC input I/O, starting at which the data are read

MD10396 $MN_PLCIO_NUM_BYTES_OUT

Number of PLC I/O output bytes that are written directly by the NC

MD10397 $MN_PLCIO_LOGIC_ADDRESS_OUT

Logical start address of the PLC output I/O, starting at which the data are written

MD10398 $MN_PLCIO_IN_UPDATE_TIME

Time period within which the data that can be read by means of $A_PBx_IN are updated. The time period is rounded up internally to the next highest multiple of the time defined by the interpolation cycle. When 0 is entered (default value), the data are updated in every interpolation cycle.

MD10399 $MN_PLCIO_TYPE_REPRESENTATION

Little-/big-endian format display of system variables $A_PBx_OUT, $A_PBx_IN for PLC I/Os that can be controlled directly from the NCK

value = 0 (Default)
System variables are displayed in little-endian format
(i.e., least significant byte at least significant address)

value = 1 (Standard format for PLC, recommended)
System variables are displayed in big-endian format
(i.e., most significant byte at least significant address)

The logical PLC I/O addresses entered in the machine data and the number of bytes to be transferred must be consistent with the PLC hardware configuration. In the configured areas, there must not be any 'address gaps' in the PLC I/O expanded configuration.

Memory organization

16 bytes (over all channels) are available for data exchange from and to the PLC respectively. These areas have to be managed by the user (that is, no overlapping of the variables, not even across channel borders).

The variables within these areas are displayed either in little-endian (= 0) or big-endian (= 1) format,
depending on the setting of the machine data:

MD10399 $MN_PLCIO_TYPE_REPRESENTATION

Since big-endian format is generally the most common display type on the PLC (that is, it also applies to the PLC I/Os), it should normally be used.

Alignment

The assignment of the input and output areas for direct PLC I/Os must satisfy the following conditions:

$A_PBB_IN[j]	; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN])
$A_PBW_IN[j]	; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN] - 1)
$A_PBD_IN[j]	; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN] - 3)
$A_PBR_IN[j]	; j < ([MD10394 $MN_PLCIO_NUM_BYTES_IN] - 3)

$A_PBB_OUT[k]	; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT])
$A_PBW_OUT[k]	; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT] - 1)
$A_PBD_OUT[k]	; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT] - 3)
$A_PBR_OUT[k]	; k < ([MD10396 $MN_PLCIO_NUM_BYTES_OUT] - 3)

Furthermore, the maximum number of bytes available for data exchange must not be exceeded.

1.2.6 Analog-value representation of the NCK analog input/output values

Conversion of analog values

The analog values are only processed by the NCU in a digital form.

Analog input modules convert the analog process signal into a digital value.

Analog output modules convert the digital output value into an analog value.

Analog-value representation

The digitized analog value is identical to input and output values with the same rating range (e.g., voltage range ±10 V DC).

The analog values are coded in the PLC interface as fixed-point numbers (16 bits including sign) in the two's complement (see table below).

Table 1- 1 Digital coding of analog values at the PLC interface

Resolution	Binary analog value															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Significance of the bits	SG	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Sign

The sign (SG) of the analog value is always located in bit 15.

Applies to SG: "0" → +
"1" → -

Resolution less than 15 bits

The analog value can be finely adjusted depending on the resolution of the digital-to-analog converter.

If the resolution of the analog module is less than 15 bits, the analog value is entered left-justified. The free least significant positions are filled with zeroes.

Using a bit pattern, the table below illustrates how zeroes are written to the free bit positions for a 14-bit and a 12-bit analog value.

With a resolution of 14 bits (including sign), the minimum increment is 1.22 mV (10 V: 8192). In this case, both least significant bits of the analog value (bit 0 and bit 1) are always 0.

With a resolution of 12 bits (including sign), the increment is 4.8 mV (10 V: 2048). Bits 0 to 3 are always 0.

Table 1- 2 Examples of digital analog-value representation

Resolution	Binary analog value															
	High byte								Low byte							
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Significance of the bits	SG	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
14-bit analog value	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	0
12-bit analog value	0	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0

For the resolutions and rating ranges of the analog input/output modules used, see:
References:
/PHD/ SINUMERIK 840D Configuration Manual NCU(HW)
/S7H/ SIMATIC S7-300 Software Installation Manual, Technology Functions.

Examples

Here are two examples of digital analog-value representation for a rating range of 10 V and 14-bit resolution.

Example 1:

Analog value:	9.5 V
Magnitude (decimal number):	7782 = j9.5(V)j:10(V) * 8192
Magnitude (pure binary number):	0111 1001 1001 10
Word (pure binary number):	0111 1001 1001 10**00**
Word (hexadecimal number):	7998

Example 2:

Analog value:	-4.12 V
Magnitude (decimal number):	3375 = j-4.12(V)j:10(V) * 8192
Magnitude (pure binary number):	0011 0100 1011 11
Two's complement:	1100 1011 0100 01
Word (pure binary number):	1100 1011 0100 01**00**
Word (hexadecimal number):	CB44

1.2.7 Comparator inputs

Function

Two internal comparator inputs bytes (with eight comparator inputs each) are available in addition to the digital and analog NCK inputs. The signal state of the comparator inputs is generated on the basis of a comparison between the analog values present at the fast analog inputs and the threshold values parameterized in setting data (see fig.).

The **$A_INCO[n]** system variable allows the signal state (i.e., the result of the comparison) of comparator input [n] to be scanned directly in the part program.

Applies to index n:

n = 1 to 8	For comparator byte 1
n = 9 to 16	For comparator byte 2

Terms

In this description, the terms "comparator inputs" (with index [n]; range of n: 1 to 8 or 9 to 16) and "comparator input bits" (with index [b]; range of b: 0 to 7) are used.

They are related as follows:

For n = 1 to 8:	Comparator input n is equivalent to comparator input bit b = n - 1.
For n = 9 to 16:	Comparator input n is equivalent to comparator input bit b = n -9.

Example

Comparator input 1 is equivalent to comparator input bit 0.

Assignment of analog inputs

The following general machine data is used to assign an analog input to input bit [b] of comparator byte 1.

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[b]

Example

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[0] = 1

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[1] = 1

MD10530 $MN_COMPAR_ASSIGN_ANA_INPUT_1[7] = 7

Analog input 1 acts on input bits 0 and 1 of comparator byte 1.

Analog input 7 acts on input bit 7 of comparator byte 1.

Similarly, the assignment for comparator byte 2 should be set using the following machine data:
MD10531 $MN_COMPAR_ASSIGN_ANA_INPUT_2[b]

Comparator parameterization

General machine data
MD10540 $MN_COMPAR_TYPE_1 is used to set the following parameters for each bit (0 to 7) of comparator byte 1:

- Comparison type mask (bits 0 to 7)

  The type of comparison conditions is defined for each comparator input bit.

Bit = 1:	Associated comparator input bit is set to "1" if the analog value ≥ the threshold value.
Bit = 0:	Associated comparator input bit is set to "0" if the analog value ≤ the threshold value.

- Output of the comparator input byte via digital NCK outputs (bits 16 to 23)

  The comparator bits can also be output directly via the digital NCK outputs byte by byte. This requires specification in this byte (bits 16 to 23) of the digital NCK output byte to be used (see MD10540 $MN_COMPARE_TYPE_1).

- Inversion mask for outputting the comparator input byte (bits 24 to 31)

  For every comparator signal it is also possible to define whether the signal state to be output at the digital NCK output is to be inverted or not.

Bit = 1:	Associated comparator input bit is not inverted.
Bit = 0:	Associated comparator input bit is inverted.

Threshold values

The threshold values used for comparisons on comparator byte 1 or 2 must be stored as setting data. For every comparator input bit [b], you must enter a separate threshold value:

SD41600 $SN_COMPAR_THRESHOLD_1[b]
(threshold values for input bit [b] of comparator byte 1); b = 0 to 7

Comparator signals as digital NCK inputs

All NC functions that are processed as a function of digital NCK inputs can also be controlled by the signal states of the comparators. The byte address for comparator byte 1 (HW byte 128) or 2 (HW byte 129) must be entered in the MD associated with the NC function ("Assignment of hardware byte used").

Example

"Multiple feedrates in one block" NC function

Entry in channel-specific machine data:

MD21220 $MC_MULTFEED_ASSIGN_FASTIN = 129

This activates various feedrate values as a function of the status of comparator byte 2.

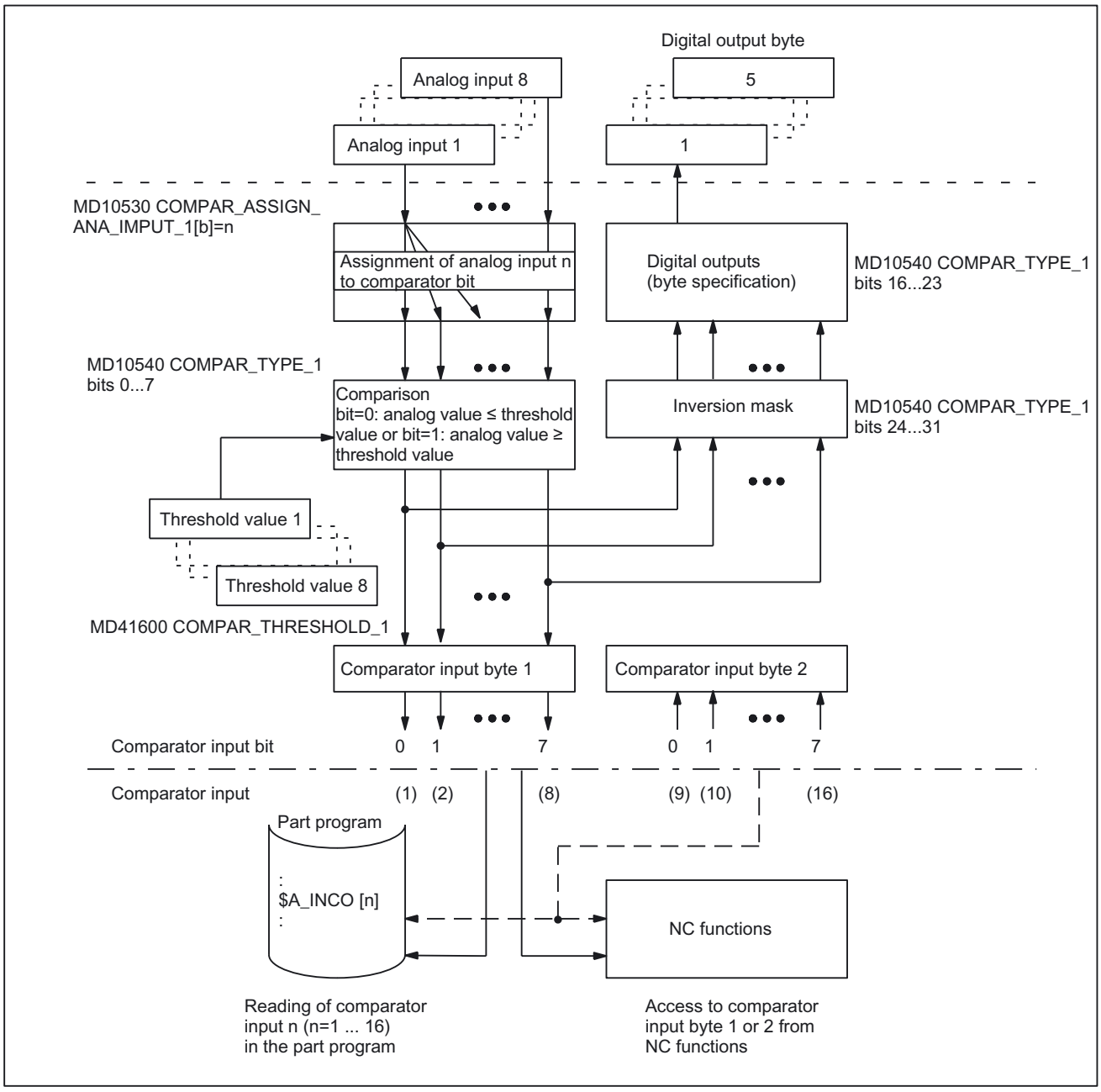


Figure 1-5 Functional sequence for comparator input byte 1 (or 2)

1.3 NCK I/O via PROFIBUS - only 840D sl

1.3.1 Functionality

General

The function "NCK-I/O via PROFIBUS" implements a direct data exchange between NCK and PROFIBUS-I/O.

The PROFIBUS-I/O is connected to the control. Like for any other PLC-I/O, an S7-HW-configuration (PLC) must be done before using this PROFIBUS-I/O.

If the individual useful-data slots of a PROFIBUS-slave are configured in the S7-HW-configuration (PLC) in such a way that they form a contiguous PROFIBUS-I/O section in series, with logical start addresses in ascending order, then this section is hereinafter referred to as I/O region.

An I/O region is characterized by:

- a logical start address (this corresponds to the logical start address of the first useful-data slot of this I/O range)
- a configured length (this corresponds to the length in bytes of the connected PROFIBUS-I/O to be accessed)

The logical start addresses of the I/O range must be available in NCK, so that it can read/write the corresponding data of the PROFIBUS-I/O via an NCK-internal PROFIBUS communication interface. The projected I/O range is registered via the machine data. The communication with the PROFIBUS-I/O is thus possible only I/O range oriented.

Data exchange

The data exchange with the PROFIBUS-I/O is done via an NCK-internal PROFIBUS-communication interface. The following options of data exchange with the PROFIBUS-I/O are available to the NCK-user:

- Reading/writing of NCK-system variables ($A_DPx_IN[n,m] or $A_DPx_OUT[n,m]) via part programs /synchronous actions in the IPO cycle (data consistency). The PROFIBUS-I/O data to be written is printed at the PROFIBUS-I/O only after the corresponding IPO cycle.
- Reading/printing the data blocks via the compile-cycle interface (data consistency for Servo cycle)

Parallel data access

A **parallel read access** through compile cycles and part programs/synchronous actions on data of the same I/O range is **possible**, as long as the corresponding I/O range has been configured for this. One must ensure that the read accesses have access to the different mappings of the PROFIBUS-I/O data. The data consistency within these mappings is ensured. However, the data equality between these mappings cannot be ensured during an IPO cycle.

A **parallel writing access** through compile cycles and part programs/synchronous actions on data of the same I/O range is **not possible**. While configuring the NCK it must be determined, whether a specific I/O range of the PROFIBUS-I/O is allocated to the system variables or to the compile cycles.

Activation

The function is activated during the NCK-ramp up.

1.3.2 Parameter assignment

The configuration of the I/O range is done via the machine data. The parameters once set can no longer be changed during the normal operation of the NCK.

16 I/O ranges in the read direction and 16 I/O ranges in the write direction are provided. NCK restricts the maximum size of the I/O ranges to 128 bytes each.

Once an I/O range is activated its availability is checked each time an IPO cycle starts. For this, the lifespan of a useful-data slot within an I/O range is evaluated. If the lifespan is not set at the start of an IPO cycle, an alarm (9050 or 9052) is issued. This alarm does not stop the processing of the part program, but instead is only displayed and remains as such till the affected I/O range gets a valid lifespan again.

Logical start address

To make specific I/O ranges available again, their logical start addresses must be conveyed to the NCK. To do this, the following machine data must be configured:

MD10500 $MN_ DPIO_LOGIC_ADDRESS_IN[n]

MD10510 $MN_ DPIO_LOGIC_ADDRESS_OUT[n]

Length of an I/O range

In order that the NCK can check, whether an I/O range has been configured completely, the expected length (bytes) of the corresponding I/O range must be entered. To do this, the following machine data must be configured:

MD10501 $MN_ DPIO_RANGE_LENGTH_IN[n]

MD10501 $MN_ DPIO_RANGE_LENGTH_OUT[n]

If the length "0" is entered, only the useful-data slot found under the corresponding logical start address is configured as I/O range. In such a case, the length of the I/O range is then compared with the length of the useful-data slot found.

Further attributes

Further attributes can be allocated to each I/O range with the following machine data:

MD10502 $MN_ DPIO_RANGE_ATTRIBUTE_IN[n]

Bit	Value	Description
0	0	Little Endian format
	1	Big Endian format
1		Reserved
2	0	Reading possible via system variables and CC-binding.
	1	Reading possible only for CC-binding.
3	0	Slot-lifespan-alarms are issued.
	1	Slot-lifespan-alarms are suppressed.

MD10512 $MN_ DPIO_RANGE_ATTRIBUTE_OUT[n]

Bit	Value	Description
0	0	Little Endian format
	1	Big Endian format
1	0	Writing only via system variable.
	1	Writing only via CC-binding.
2		Reserved
3	0	Slot-lifespan-alarms are issued.
	1	Slot-lifespan-alarms are suppressed.

Note

- The configuration of the I/O ranges via the corresponding machine data need not be done persistently. That is, the assignment of the I/O ranges to the corresponding machine data-indices can be selected in any way.
- If the registration of one/several I/O range(s) does not run successfully during the NCK-ramp up phase, the registration is aborted with an alarm (4700 or 4702).

1.3.3 Programming

Requirement

- Correct configuration of the corresponding I/O ranges.
- PLC must actually be able to provide the required I/O ranges (useful-data slots).
- The configured I/O ranges are released for use only when the PROFIBUS-communication interface is able to do a data exchange with the corresponding PROFIBUS-I/O for the first time.

1.3.3.1 Communication via part programs/synchronous actions

General

The NCK global system variables enable access to the PROFIBUS-I/O, whether read or write, from the part programs/synchronous actions:

- $A_DPx_IN [n,m]
- $A_DPx_OUT [n,m]

Please note the following:

- When reading from or writing to these variables from the part programs, a preprocessing stop is triggered.
- To ensure data consistency during programming from the part program and the synchronous actions, the PROFIBUS-I/O data are accessed, which are kept consistent for the respective IPO cycle.
- If the same PROFIBUS-I/O data are to be write-accessed several times within an IPO cycle (e.g. synchronous actions, access from different channels, etc.), then the data of the last write access respectively are valid.
- The PROFIBUS-I/O data to be written is printed at the PROFIBUS-I/O only after the corresponding IPO cycle.

Access I/O range data

The following system variables are available for accessing the I/O range data:

Table 1- 3 NCK → PROFIBUS-I/O

System variables	Value	Description
$A_DPB_OUT[n,m]	8 bit unsigned	Writing a data byte (8 bit) to PROFIBUS-IO
$A_DPW_OUT[n,m]	16 bit unsigned	Writing a data word (16 bit) to PROFIBUS-IO
$A_DPSB_OUT[n,m]	8 bit signed	Writing a data byte (8 bit) to PROFIBUS-IO
$A_DPSW_OUT[n,m]	16 bit signed	Writing a data word (16 bit) to PROFIBUS-IO
$A_DPSD_OUT[n,m]	32 bit signed	Writing a data double word (32 bit) to PROFIBUS-IO
$A_DPR_OUT[n,m]	32 bit REAL	Writing output data (32 bit REAL) to PROFIBUS-IO

n = index for the output data range; m = byte index for the data

Table 1- 4 PROFIBUS-I/O → NCK

System variables	Value	Description
$A_DPB_IN[n,m]	8 bit unsigned	Reading a data byte (8 bit) from PROFIBUS-IO
$A_DPW_IN[n,m]	16 bit unsigned	Reading a data word (16 bit) from PROFIBUS-IO
$A_DPSB_IN[n,m]	8 bit signed	Reading a data byte (8 bit) from PROFIBUS-IO
$A_DPSW_IN[n,m]	16 bit signed	Reading a data word (16 bit) from PROFIBUS-IO
$A_DPSD_IN[n,m]	32 bit signed	Reading a data double word (32 bit) from PROFIBUS-IO
$A_DPR_IN[n,m]	32 bit REAL	Reading of output data (32 bit REAL) from PROFIBUS-IO

n = index for the input data range; m = byte index for the data

Check configuration of the I/O ranges

The configuration of the I/O ranges can be checked via the following system variables. Each bit of these bit-fields correspond to an I/O range. It is set, when the I/O range is configured for access via the part programs/synchronous actions.

System variables	Value	Description
$A_DP_IN_CONF	32 bit bit-field	Reading all configured input data ranges of the PROFIBUS-IO
$A_DP_OUT_CONF	32 bit bit-field	Reading all configured output data ranges of the PROFIBUS-IO

Check availability of the I/O ranges

The availability of the I/O ranges can be checked via the following system variables. Each bit of these bit-fields correspond to an I/O range. It is set, when the I/O range is ready for access via the part programs/synchronous actions.

System variables	Value	Description
$A_DP_IN_VALID	32 bit bit-field	Reading all valid input data ranges of the PROFIBUS-IO
$A_DP_OUT_VALID	32 bit bit-field	Reading all valid output data ranges of the PROFIBUS-IO

Query status of an I/O range

The exact status of an I/O range can be queried with the help of the following system variables.

System variables	Value	Description
$A_DP_IN_STATE[n] n = index for the input data range	0: Data range was not configured 1: Data range could not be activated 2: Data range is available 3: Data range is currently not available	Reading the state of the input data range
$A_DP_OUT_STATE[n] n = index for the output data range		Reading the state of the output data range

Query length of an I/O range

The configured length an I/O range can be queried with the help of the following system variables.

System variables	Description
$A_DP_IN_LENGTH[n] n = index for the input data range	Reading the length of the input data range
$A_DP_OUT_LENGTH[n] n = index for the output data range	Reading the length of the output data range

Note

- Via <n> (RangeIndex) the corresponding NCK-configured E/A-range is selected. If the required I/O range is not configured, it is indicated by issuing an alarm (17020).
- The <m> (RangeOffset) points to the place (byte-offset) within the I/O range, from which the data access is to be started. Data types can be read/written at any byte-offset within the I/O range. Read/write accesses, which exceed the configured limits of the respective I/O range, are rejected with the generation of an alarm (17030).
- Via the machine data:
  MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN or
  MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT
  the display format (Little-/Big-Endian) for $A_DPx_IN[n,m]- or $A_DPx_OUT[n,m]-system variables can be defined for read/write direction as well as for each individual I/O range.

1.3.3.2 Communication via compile cycles

General

The CC-bindings are available for reading/printing the data blocks via the compile cycle interfaces. The access to the data of the I/O range takes place at the Servo-task level. The data are updated in each Servo cycle.

Data consistency is thus given for each respective Servo cycle.

To have a write access to the data of the I/O range via the CC-bindings, the respective I/O ranges must be cleared during the NCK-configuration by the machine data:
MD10512 $MN_DPIO_SLOT_ATTRIBUTE_OUT (attributes of the PROFIBUS-I/Os)
for the programming via compile cycles.

A simultaneous programming of these I/O ranges via part programs/synchronous actions is prevented by issuing an alarm (17020).

It must be noted that the data is displayed in general in the PLC in the Big-Endian format. Naturally, this also applies to the PROFIBUS-I/O. Since the bindings support only the byte-oriented access to data ranges (byte-offset, number of bytes to be transmitted) within an I/O range, you must pay attention to the correct display of the data types (16 bit, 32 bit, etc.).

CC-Bindings

The following CC-bindings are available:

CCDataOpi: getDpIoRangeConfiguration()

CCDataOpi: getDpIoRangeValid()

CCDataOpi: getDpIoRangeInInformation()

CCDataOpi: getDpIoRangeOutInformation()

CCDataOpi: getDpIoRangeInState()

CCDataOpi: getDpIoRangeOutState()

CCDataOpi: getDataFromDpIoRangeIn()

CCDataOpi: putDataToDpIoRangeOut()

Note

- The bindings
  CCDataOpi: getDataFromDpIoRangeIn() or
  CCDataOpi: putDataToDpIoRangeOut()
  monitor during the read/write accesses the adherence to the limits of the respective I/O range configured at the NCK and PLC-side. Access to data/data ranges, which do not lie completely within the configured I/O range limits, are rejected by returning the enumerator CCDATASTATUS_RANGE_LENGTH_LIMIT.
- If an attempt is made to access an I/O range which is not configured (or not configured for the compile cycle), it is notified by means of the return-enumerator CCDATASTATUS_RANGE_NOT_AVAILABLE.

NOTICE
The compile cycle programmer himself is responsible for the correct use of the CC-bindings! It must be noted that the additional performance requirement needed for providing the data of the configured I/O ranges at the Servo-task level, does not lead to a Servo-level computing time overflow. See the OEM documents for more information about the use of these bindings.

1.4 Constraints

1.4.1 NCK I/O via PLC

Availability of the function "digital and analog NC inputs/outputs"

Digital and analog CNC inputs/outputs (DI, DO, AI, AO) are available as follows:

- SINUMERIK 840D with NCU 571

  4 DI/4 DO (on board)

  32 DI/32 DO with expansion via NCU terminal block
- SINUMERIK 840D with NCU 572/573, SW 2 and higher

  4 DI/4 DO (on board)

  32 DI/32 DO and 8AI/8AO with expansion via NCU terminal block

Analog I/Os for 840Di

The analog I/Os are connected to the SINUMERIK 840Di via PROFIBUS-DP.

Configuration

- If the PLC I/Os are to be written/read via the fast data channel, they must always be configured as a cohesive block (i.e., no address gaps within this block).
- It must be possible for the number of bytes that have to be transferred to be mapped without gaps on the PLC I/Os.

Dynamic response

The time when the data are read in from the PLC I/Os is not synchronized with the time when the data are made available to the part program via system variables!

Data transfer (NCK <--> PLC)

- The data buffer is always output complete to the PLC I/Os, even if only one system variable was assigned within the data buffer.
- If values are assigned to several system variables simultaneously (e.g., for initializing the PLC I/Os), there is no guarantee that they will be transferred in the same interpolation cycle.

1.4.2 NCK I/O via PROFIBUS - only 840D sl

System

The function is available in the SINUMERIK 840D sl system for isochronous and non-isochronous configured PROFIBUS-I/Os.

Hardware

- The required PROFIBUS-I/O must be available and ready to use.
- A correct S7-HW-configuration (PLC-side) must be done of the required PROFIBUS-I/O and loaded in the PLC.
- An I/O range must be present on the same PROFIBUS-slave.
- Only the PROFIBUS-slaves at the first real PROFIBUS-strand of the PLC (plug with the identification DP1) are supported.

Software

- The NCK must be configured correctly via the corresponding machine data.
- A simultaneous writing of the PROFIBUS-I/O from a PLC-application program is not permitted and cannot be prevented technically.
- I/O ranges for the write access (MD10510 $MN_DPIO_LOCIG_ADDRESS_OUT[]) to the PROFIBUS-I/O may not lie in the I/O mapping range of the PLC (e.g. PLC 317, addresses from 0 - 255), since this range is used by the PLC-operating system.

1.5 Examples

1.5.1 NCK I/O via PLC

1.5.1.1 Writing to PLC-I/Os

The following assumptions are made in this example:

- Data are to be output directly to the following PLC I/Os:

  - log. addr. 521: ;8-bit digital output module
  - log. addr. 522: ;16-bit digital output module

- $A_PBx_OUT is used to output the data from **synchronized actions**.

Parameter assignment

The machine data should be set as follows:

MD10397 $MN_PLCIO_LOGIC_ADRESS_OUT= 521	;Data are output from log. addr. 521 onwards
MD10396 $MN_PLCIO_NUM_BYTES_OUT= 3	;A total of 3 bytes have to be output
MD10399 $MN_PLCIO_TYPE_REPRESENTATION = 1	;Data are displayed in big-endian format

Booting of NCK and PLC

Once the NCK and PLC have booted, there is **no** cyclic data transfer to the PLC I/Os (for write access).

Programming

Loading and starting of the part program with the following content:

```
...
ID = 1 WHENEVER TRUE DO $A_PBB_OUT[0] = 123                ;Cyclic output
                                                           ;(per interpolation
                                                           cycle)
...
ID = 2 WHEN $AA_IW[x] >= 5 DO $A_PBW_OUT[1] = 'Habcd'      ;Output of a
                                                           ;hex value
...
```

1.5.1.2 Reading from PLC-I/Os

The following assumptions are made in this example:

- PLC I/Os:

  - log. addr. 420: 16-bit analog input module
  - log. addr. 422: 32-bit digital input module
  - log. addr. 426: 32-bit DP slave input
  - log. addr. 430: 8-bit digital input module

- $A_PBx_IN is used to read in data from a part program into R parameters.
- In order to avoid slowing down the PLC user program unnecessarily (OB1), an update time (for read access) was configured in machine data
  MD10398 $MN_PLCIO_IN_UPDATE_TIME
  such that an update is only performed in every third interpolation cycle.

Parameter assignment

The machine data should be set as follows:

MD10395 $MN_PLCIO_LOGIC_ADRESS_IN = 420	;Data are read in from log. addr. 420 onwards
MD10394 $MN_PLCIO_NUM_BYTES_IN = 11	;A total of 11 bytes have to be read in
MD10398 $MN_PLCIO_IN_UPDATE_TIME = 0.03	;Update time period = 30 ms (interpolation cycle = 12 ms)
MD10399 $MN_PLCIO_TYPE_REPRESENTATION = 1	;Data are displayed in big-endian format

Booting of NCK and PLC:

The update (for read access) is now performed in every third interpolation cycle after the NCK and PLC have booted.

Programming

Loading and starting of the part program with the following content:

```
...
R1 = $A_PBW_IN[0]          ;Read in 16-bit integer
R2 = $A_PBD_IN[2]          ;Read in 32-bit integer
R3 = $A_PBR_IN[6]          ;Read in 32-bit float
R4 = $A_PBB_IN[10]         ;Read in 8-bit integer
...
```

1.5.2 NCK I/O via PROFIBUS - only 840D sl

1.5.2.1 PROFIBUS-I/O in write direction

Requirement

The S7-HW-configuration is already done.

Configuration for programming via part program/synchronous actions

- RangeIndex = 5 (NCK-internal configuration)
- as per S7-HW-configuration:
  - Log. start address = 334
  - Slot length = 8 byte
  - To be displayed in Little-Endian format

This results in the following configuration of the machine data:

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[5] = 334 (log. start address I/O range)

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[5] = 8 (length of the I/O range)

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[5]
Bit0 = 0 (Little-Endian-Format)
Bit1 = 0 (writing only via system variable)
Bit3 = 0 (Slot-lifespan-alarms issued)

Configuration for programming via CompileCycles

- RangeIndex = 6 (NCK-internal configuration)
- as per S7-HW-configuration:
  - Log. start address = 444
  - Slot length = 10 byte
  - To be displayed in Little-Endian format

This results in the following configuration of the machine data:

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[6] = 444 (log. start address I/O range)

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[6] = 0
(a single useful-data slot is to be used)

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[6]
Bit0 = 0 (Little-Endian-Format)
Bit1 = 1 (writing only via CC-binding)
Bit3 = 1 (Slot-lifespan-alarms suppressed)

Programming

```
$A_DPB_OUT[5,6]=128          ; write (8 bit) to RangeIndex=5, RangeOffset=6
$A_DPW_OUT[5,5]='B0110'      ; write (16 bit) to RangeIndex=5, RangeOffset=5
                             ; Little-Endian-format
                             ; Caution: RangeData of byte 6 are overwritten
$A_DPSD_OUT[5,3]='8FHex'     ; write (32 bit) to RangeIndex=5, RangeOffset=3
                             ; Little-Endian-format
                             ; Caution: RangeData of byte 5.6 are overwritten
$AC_MARKER[0]=5
$AC_MARKER[1]=3
$A_DPSD_OUT[$AC_MARKER[0],$AC_MARKER[1]]='8FHex'
                             ; write (32 bit) to RangeIndex=5, RangeOffset=3
                             ; Little-Endian-format
                             ; indirect programming
R1=$A_DPB_OUT[5,6]           ; read (8 bit) on RangeIndex=5, RangeOffset=6
                             ; Little-Endian-format
                             ; Result: 0xFF
ID=1 WHENEVER TRUE DO $A_DPB_OUT[5,0]=123
                             ; Cyclic output;(per IPO cycle)
$A_DPB_OUT[5.255]=128        ; Alarm 17030 because the RangeOffset is invalid.
$A_DPB_OUT[6.10]=128         ; Alarm 17020 because this range of part program
                             ; cannot be written.
$A_DPB_OUT[7.10]=128         ; Alarm 17020 because this range is not defined.
$A_DPB_OUT[16.6]=128         ; Alarm 17020 because RangeIndex >= max. available
                             ; number of ranges.
```

1.5.2.2 PROFIBUS-I/O in read direction

Requirement

The S7-HW-configuration is already done.

Configuration for programming via part program/synchronous actions

- RangeIndex = 0 (NCK-internal configuration)
- as per S7-HW-configuration:
  - Log. start address = 456
  - Slot length = 32 byte
  - To be displayed in Big-Endian format

This results in the following configuration of the machine data:

MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[0] = 456 (log. start address I/O range)

MD10501 $MN_DPIO_RANGE_LENGTH_IN[0] = 32 (length of the I/O range)

MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN[0]
Bit0 = 1 (Big-Endian-Format)
Bit2 = 0 (read possible via system variable and CC-binding)
Bit3 = 0 (Slot-lifespan-alarms issued)

Configuration for programming via CompileCycles

- RangeIndex = 1 (NCK-internal configuration)
- as per S7-HW-configuration:
  - Log. start address = 312
  - Slot length = 32 byte
  - To be displayed in Little-Endian format

This results in the following configuration of the machine data:

MD10500 $MN_DPIO_LOGIC_ADDRESS_IN[1] = 312 (log. start address I/O range)

MD10501 $MN_DPIO_RANGE_LENGTH_IN[1] = 32 (length of the I/O range)

MD10502 $MN_DPIO_RANGE_ATTRIBUTE_IN[1]
Bit0 = 1 (Big-Endian-Format)
Bit2 = 1 (read possible only via CC-binding)
Bit3 = 1 (Slot-lifespan-alarms suppressed)

Programming

```
$AC_MARKER[0]=$A_DPW_IN[0,0]       ; read (16 bit) on RangeIndex=0, RangeOffset=0
                                   ; Big-Endian-format
$AC_MARKER[1]=$A_DPSD_IN[0,1]      ; read (32 bit) on RangeIndex=0, RangeOffset=1
                                   ; Big-Endian-format
$AC_MARKER[1]=$A_DPSD_IN[0,8]      ; read (32 bit) on RangeIndex=0, RangeOffset=8
                                   ; Big-Endian-format
$AC_MARKER[2]=0
$AC_MARKER[3]=8
$AC_MARKER[1]=$A_DPSD_IN[$AC_MARKER[2],$AC_MARKER[3]]
                                   ; read (32 bit) on RangeIndex=0, RangeOffset=8
                                   ; Big-Endian-format
                                   ; indirect programming
ID=2 WHEN $A_DPB_IN[0,11]>=5 DO $AC_MARKER[2]='ABCDHex'
                                   ; Cyclic reading;(per IPO cycle)
R1=$A_DPB_IN[0,255]                ; Alarm 17030 because the RangeOffset is invalid.
R1=$A_DPB_IN[2,6]                  ; Alarm 17020 because this range is not defined.
R1=$A_DPB_IN[1,10]                 ; Alarm 17020 because this range of part program
                                   ; cannot be written.
R1=$A_DPB_IN[16,6]                 ; Alarm 17020 because RangeIndex >= max.
                                   ; available number of ranges.
```

1.5.2.3 Query of the RangeIndex in case of "PROFIBUS-I/O in write direction"

Requirement

The S7-HW-configuration is already done.

Configuration for programming via part program/synchronous actions

- RangeIndex = 5 (NCK-internal configuration)
- as per S7-HW-configuration:
  - Log. start address = 1200
  - Slot length = 32 byte
  - To be displayed in Big-Endian format

This results in the following configuration of the machine data:

MD10510 $MN_DPIO_LOGIC_ADDRESS_OUT[5] = 1200 (log. start address I/O range)

MD10511 $MN_DPIO_RANGE_LENGTH_OUT[5] = 0
(a single useful-data slot is to be used)

MD10512 $MN_DPIO_RANGE_ATTRIBUTE_OUT[5]
Bit0 = 1 (Big-Endian-Format)
Bit1 = 0 (writing only via system variable)
Bit3 = 0 (Slot-lifespan-alarms issued)

Programming

```
before an access query the status of RangeIndex = 5
N3      check:                                      ; Jump marker
N5      IF $A_DP_OUT_STATE[5]==2 GOTOF write        ; if data range valid
                                                    ; => jump to N15
N10     GOTOB check                                 ; jump back to check
N15     write:                                      ; Jump marker
N20     $A_DPB_OUT[5,6]=128                         ; Writing the data byte
```

```
Query, whether all configured ranges/slots are valid
N3      check:                                      ; Jump marker
N5      IF $A_DP_OUT_CONF==$A_DP_OUT_VALID GOTOF    ; if data range valid
        write                                       ; => jump to N15
N10     SETAL(61000)                                ; Set alarm no. 61000
N15     write:                                      ; Jump marker
N20     $A_DPB_OUT[5,6]=128                         ; Writing the data byte
```

Query, whether the configured RangeIndex = 5 is valid

```
N3      check:                                      ; Jump marker
N5      IF $A_DP_OUT_VALID B_AND 'B100000' GOTOF    ; if data range valid
        write                                       ; => jump to N15
N10     SETAL(61000)                                ; Set alarm no. 61000
N15     write:                                      ; Jump marker
N20     $A_DPB_OUT[5,6]=128                         ; Writing the data byte
```

Querying the length of the configured, valid I/O range with RangeIndex = 5

```
N100    R1=$A_DP_OUT_LENGTH[5]          ; Length of the I/O range (slot) in bytes
                                        ; Result: R1 = 32
```

1.6 Data lists

1.6.1 Machine data

1.6.1.1 General machine data

Number	Identifier: $MN_	Description
10300	FASTIO_ANA_NUM_INPUTS	Number of active analog NCK inputs
10310	FASTIO_ANA_NUM_OUTPUTS	Number of active analog NCK outputs
10320	FASTIO_ANA_INPUT_WEIGHT	Weighting factor for analog NCK inputs
10330	FASTIO_ANA_OUTPUT_WEIGHT	Weighting factor for analog NCK outputs
10350	FASTIO_DIG_NUM_INPUTS	Number of active digital NCK input bytes
10360	FASTIO_DIG_NUM_OUTPUTS	Number of active digital NCK output bytes
10362	HW_ASSIGN_ANA_FASTIN	Hardware assignment of external analog NCK inputs
10364	HW_ASSIGN_ANA_FASTOUT	Hardware assignment of external analog NCK outputs
10366	HW_ASSIGN_DIG_FASTIN	Hardware assignment of external digital NCK inputs
10368	HW_ASSIGN_DIG_FASTOUT	Hardware assignment of external digital NCK outputs
10380	HW_UPDATE_RATE_FASTIO	Update rate of clock-synchronous external NCK I/Os
10382	HW_LEAD_TIME_FASTIO	Rate time for clock-synchronous external NCK I/Os
10384	HW_CLOCKED_MODULE_MASK	Clock-synchronous processing of external NCK I/Os
10394	PLCIO_NUM_BYTES_IN	Number of directly readable input bytes of the PLC I/Os
10395	PLCIO_LOGIC_ADDRESS_IN	Start address of the directly readable input bytes of the PLC I/Os
10396	PLCIO_NUM_BYTES_OUT	Number of directly writeable output bytes of the PLC I/Os
10397	PLCIO_LOGIC_ADDRESS_OUT	Start address of the directly writeable output bytes of the PLC I/Os
10398	PLCIO_IN_UPDATE_TIME	Update time for PLC-I/O input cycle
10399	PLCIO_TYPE_REPRESENTATION	Little-/big-endian representation for PLC I/O
10500	DPIO_LOGIC_ADDRESS_IN	logical slot address of the PROFIBUS-I/O
10501	DPIO_RANGE_LENGTH_IN	Length of the PROFIBUS-I/O range
10502	DPIO_RANGE_ATTRIBUTE_IN	Attributes of the PROFIBUS-I/O
10510	DPIO_LOGIC_ADDRESS_OUT	Logical slot address of the PROFIBUS-I/O
10511	DPIO_RANGE_LENGTH_OUT	Length of the PROFIBUS-I/O range
10512	DPIO_RANGE_ATTRIBUTE_OUT	Attributes of the PROFIBUS-I/O
10530	COMPAR_ASSIGN_ANA_INPUT_1	Hardware assignment of NCK analog inputs for comparator byte 1
10531	COMPAR_ASSIGN_ANA_INPUT_2	Hardware assignment of NCK analog inputs for comparator byte 2
10540	COMPAR_TYPE_1	Parameterization for comparator byte 1
10541	COMPAR_TYPE_2	Parameterization for comparator byte 2

1.6.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
21220	MULTFEED_ASSIGN_FASTIN	Assignment of input bytes of NCK I/Os for "multiple feedrates in one block"

1.6.2 Setting data

1.6.2.1 General setting data

Number	Identifier: $SN_	Description
41600	COMPAR_THRESHOLD_1	Threshold values for comparator byte 1
41601	COMPAR_THRESHOLD_2	Threshold values for comparator byte 2

1.6.3 Signals

1.6.3.1 Signals to NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Disable digital NCK inputs	DB10.DBB0/122/124/126/128	DB2800.DBB0/1000
Setting on PLC of digital NCK inputs	DB10.DBB1/123/125/127/129	DB2800.DBB1/1001
Disable digital NCK outputs	DB10.DBB4/130/134/138/142	DB2800.DBB4/1008
Overwrite mask for digital NCK outputs	DB10.DBB5/131/135/139/143	DB2800.DBB5/1009
Setting value from PLC for the digital NCK outputs	DB10.DBB6/132/136/140/144	DB2800.DBB6/1010
Setting mask for digital NCK outputs	DB10.DBB7/133/137/141/145	DB2800.DBB7/1011
Disable analog NCK inputs	DB10.DBB146	-
Setting mask for analog NCK inputs	DB10.DBB147	-
Setting value from PLC for the analog NCK inputs	DB10.DBB148-163	-
Overwrite mask for analog NCK outputs	DB10.DBB166	-
Setting mask for analog NCK outputs	DB10.DBB167	-
Disable analog NCK outputs	DB10.DBB168	-
Setting value from PLC for the analog NCK outputs	DB10.DBB170-185	-

1.6.3.2 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Actual value for digital NCK inputs	DB10.DBB60/186-189	DB2900.DBB0/1000
Setpoint for digital NCK outputs	DB10.DBB64/190-193	DB2900.DBB4/1004
Actual value for analog NCK inputs	DB10.DBB194-209	-
Setpoint for analog NCK outputs	DB10.DBB210-225	-

B3: Several operator panels connected to several NCUs, distributed systems - only 840D sl

2

2.1 Brief description

2.1.1 Topology of distributed system configurations

Features

Rotary indexing machines, multi-spindle turning machines and complex NC production centers all exhibit one or more of the following features:

- More than one NCU due to large number of axes and channels
- Large dimensions and when spatially separated require more operator units (operator panel fronts OP/TP with PCU/TCU, machine control panel MCP, handheld terminal HT8)
- Modular machine concept, e.g. using distributed control cabinets

Distributed system configuration

The two areas highlighted in the topology display identify two communications functions to be examined separately in terms of configuration and utilization.

An operator panel generally comprises an OP/TP with a PCU on which the HMI software runs. If several OPs/TPs are to be connected to a PCU, TCUs are required in addition.

Reference:
Operator Components and Networking Manual

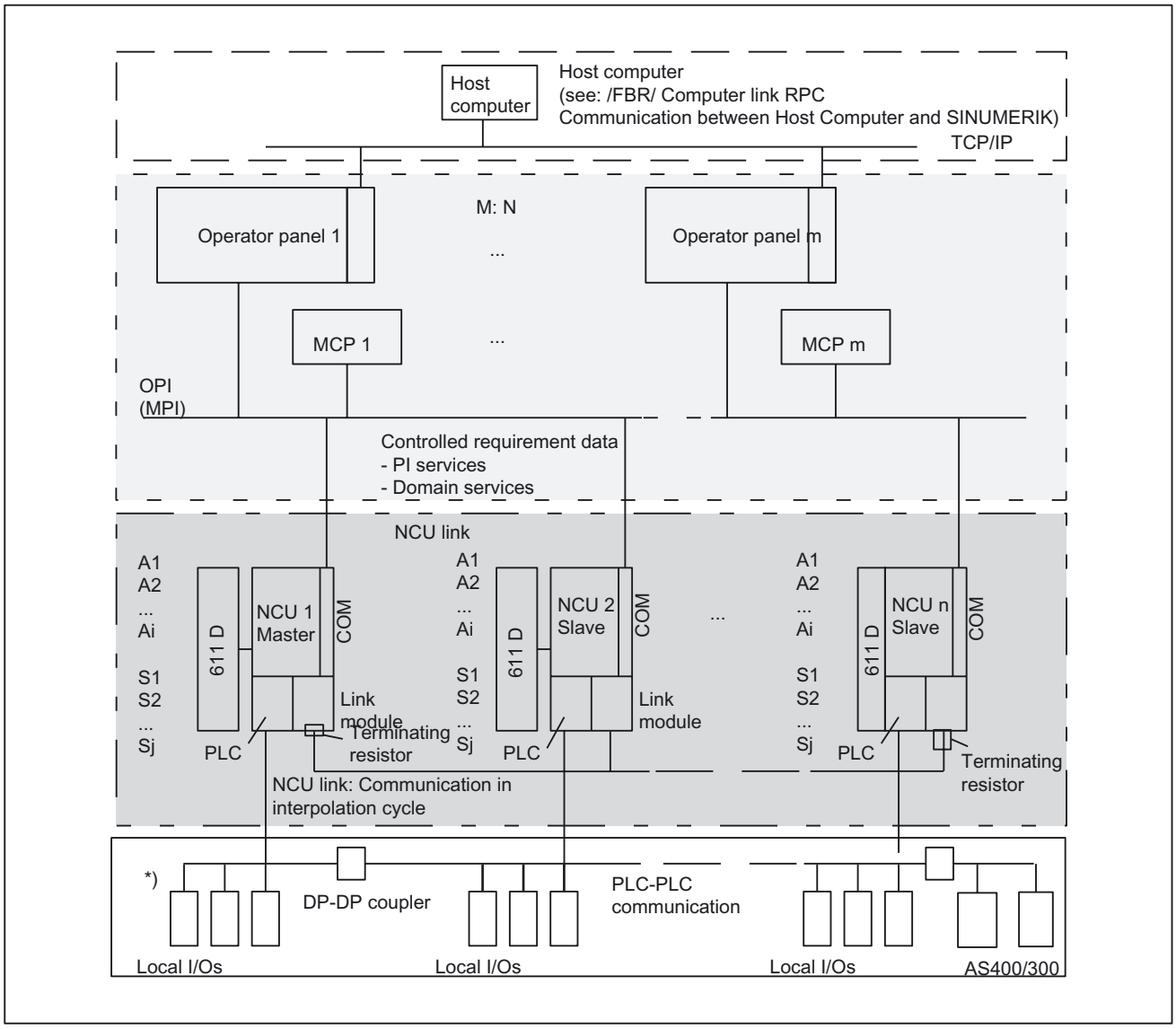


Figure 2-1 Topology of distributed system configurations

*) PLC-PLC communication entails one of the following:
- PLC-PLC cross-communication master, slave comm.)
or
- Local PLC I/Os

M: N

Assignment of several control units (M) to several NCUs (N):

- Bus addresses, bus type
- Properties of the control units:
  - Main control panel/secondary control panel
- Dynamic switchover from PCUs/MCPs or HT6s to other NCUs

Actions are required for the use of M:N during:

- Hardware configuration

  Reference:
  Guidelines for Machine Configuration
- File parameterization
- Design of the PLC program

  Reference:
  Function Manual Basic Functions; PLC Basic Program (P3)
- Operation

  Reference:
  Operating Manual

For applications/configurations matching the examples described, the notation examples can be copied directly or modified slightly. The aspects file parameterization, PLC programming and operation are described as an aid to quick commissioning in each case.

NCU link

The functions for the NCU link are based on additional communication between NCUs in the interpolation cycle.

The NCU link allows:

- Subordination of a physical axis to several different NCUs
- Cross-NCU interpolation
- An increase in the number of usable axes for an NCU grouping
- An increase in the number of channels for an NCU grouping
- Provision of axis data and signals on the NCU to which a non-local axis is temporarily assigned
- User communication via the NCU grouping by means of link variables

Lead link axes

Following axes can be traversed by an NCU if the associated leading axis is being traversed by another NCU. The NCU link communication handles the necessary exchange of axis data.

NCU link with different IPO cycles

It is possible to use an NCU link connection between NCUs with different interpolation cycles for specific applications, e.g. non-circular turning.

Host computer

Communication between master computers and operator panels is described in:
Reference:
Description of Functions Computer Coupling RPC SINUMERIK

PLC-PLC communication

DP Master, DP Slave, DP-DP coupler, cross-communication via PBK

Bus capacities

The buses illustrated in the diagram above are specially designed for their transmission tasks. The resultant communication specifications are shown in the next diagram:

- Number of bus nodes
- Baud rate
- Synchronization

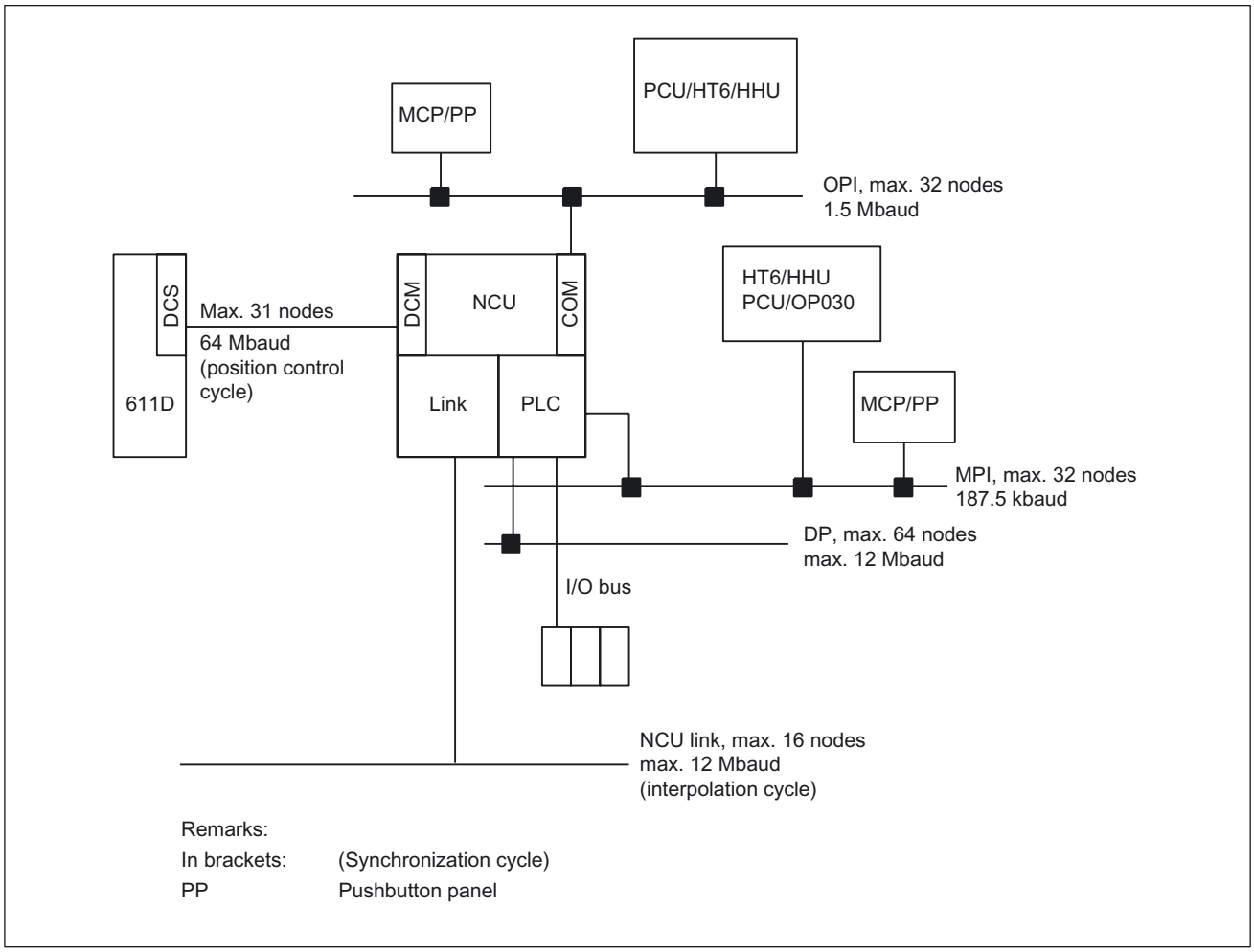


Figure 2-2 Bus properties

7-layer model structure

Communication takes place on the following protocol layers:

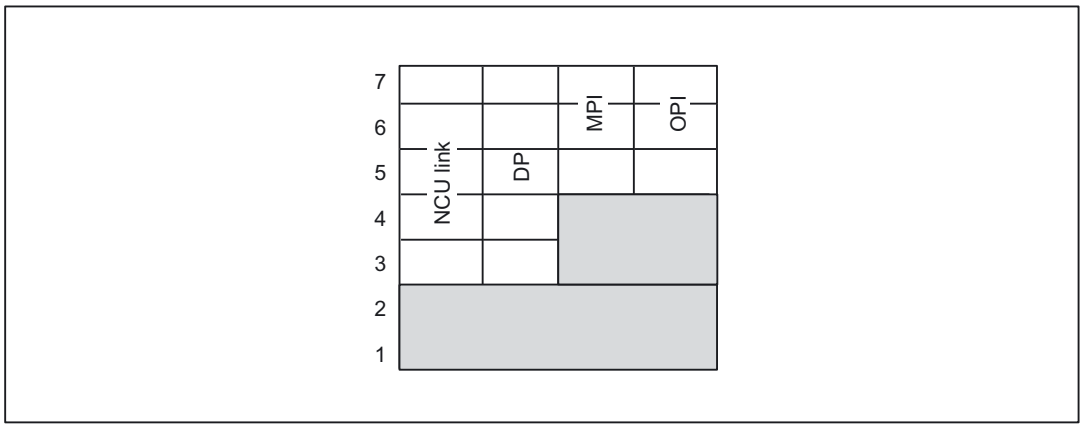


Figure 2-3 Protocol levels of 7-layer model

The NCU link and DP can operate faster because they are assigned directly to layer 2.

2.1.2 Several operator panels and NCUs with control unit management (option)

2.1.2.1 General information

Introduction

The system configuration must be highly flexible so as to meet the requirements of complex machines, such as rotary indexing machines, multi-spindle machines and complex NC production centers.

They frequently require:

- Several control units (M) due to large machine dimensions and physical separation of operator stations
- Several NCUs (N) due to large numbers of axes and channels

Restriction

The standard functionality applies to all SW versions without the option. However different performance grades depending on SW version must also be taken into account (see "Supplementary conditions").

While the standard functionality supports only certain restricted M:N combinations, the "control unit management" option provides a flexible, universal solution for satisfying the requirements above.

2.1.2.2 System features

M:N concept

This concept allows the user to connect any control units to any NCUs in the system (within the limits imposed by the hardware) via the bus and to switch them over as and when required.

NCU link

NCU link is an additional direct connection between the NCUs which enables fast communication.

New features

New features connected with the"control unit management" option are as follows:

- Independent connection of PCU and MCP
- Two independent HMI connections for each NCU:
  - PCU and MCP can be switched over together, or the PCU on its own.
  - HMI states:
    - online/active: Operation and monitoring
    - online/passive: A window is displayed with header and alarm line and a message indicating the "passive" state
    - offline
- Different bus systems (MPI/OPI) between PCU/MCP/HT6 and NCUs

  (Changes do not take effect until after power up.)
- HMI function as server/as main, secondary control panel
- A combination of both fixed and switchable operator panels can be connected.
- Suppression mechanism (priority-controlled) if more than two PCUs are competing for an NCU connection
- Up to 32 bus nodes (PCUs, MCPs, HT6s and/or NCUs)
- PLC controls the switchover process (control unit switchover to toolbox, directory PSP_PROG\m_zu_n.zip)
- Configuration file NETNAMES.INI with new parameters

Supplementary conditions

At any one time:

- A max. of two PCUs/HT6s can be online on one NCU.
- Only one of them can be in the active state.
- The same value must be entered in HT6 for the PCU and the MCP address (because the MCP addresses can only accept values up to 15, the PCU address is limited).

2.1.2.3 Hardware

Operator panel fronts

The OP/TP operator panel fronts incorporate a slimline screen, softkeys, a keyboard, interfaces and a power supply.

Machine control panel

The machine control panel (MCP) incorporates a keyboard, a rotary button pad and interfaces.

HT6

Handheld Terminal HT6 incorporates a slimline screen, softkeys, a keyboard, an override rotary switch, emergency stop and enabling buttons, as well as interfaces and a power supply. The functions of both the operator panel and the MCP are integrated in the HT6.

Difference between OP030 and HT6

OP030 and HT6 differ with respect to their NCU assignment options:

- OP030

  Can only be permanently assigned to one NCU. It can be used as a second operator panel front for this NCU. The addresses of the connected partners can be set for this purpose.

- HT6

  Can be actively assigned to another NCU via HT6/HMI operation.

References

The operator interfaces are described in the operating manuals for the operator panel fronts used:

/FBO/ Function Manual, Configuration Operator Interface OP030

/BHsl/ Operator Components Manual

Buses

The control units (PCUs and/or MCPs, HT6s) and the NCUs are connected via:

- MPI bus (Multi-point interface; 187.5 kbaud)

or

- OPI bus (operator panel front interface, 1.5 Mbaud).

It is possible to combine different bus systems within one installation.

Address assignments

Bus nodes each have a unique address on the bus.

The NCU uses:

- A common address for the NC and PLC on the **OPI**
- Two separate addresses (for NC and PLC) on the **MPI interface**

  The following applies:

  – The PLC address can be reconfigured with STEP7.
  "2" is the default address for the PLC on the MPI.

  – The following applies for the addresses on the MPI interface:
  NC address = PLC address + 1

Defaults for OPI

Addresses 0 (PG diagnostics) and 14 (HT6) are reserved at the factory and 13 (NCK) is defined as the default. These addresses should not be assigned to bus nodes in M:N systems.

- **Address 0**

  Is reserved for PG diagnostics.

  Note

  The M:N switchover can malfunction when a PG is online.

  Remedy:

  Either set the PG to offline before switching the unit over or connect it to the MPI interface.

- **Address 13**
  Is defined as the default for service/commissioning.

  It is possible to reconfigure this address via HMI operation. Reserve the address for the "NCU replacement" service case if possible.

Defaults for MPI

- **Address 2**

  For PLC

- **Address 3**

  For NCU

Number of active PCUs/HT6s on 1 NCU

A maximum of two PCUs/HT6s (incl. COROS OPs) can be constantly actively connected to one NCU. PCUs/HT6s on the OPI or MPI count in the same way.

Number of MCPs/HHUs on 1 NCU

Two MCPs and one HHU can be connected to the OPI or MPI interface of an NCU as standard.

Note

The MPI/OPI network rules outlined in the "SINUMERIK 840D Commissioning Manual" must be applied.

In particular, an M:N system must be connected up by means of cables fitted with terminating resistors (identifiable by switch with which these are switched in and out).

2.1.2.4 Functions

Defining properties

The assignment of HMI properties can be **static** or **dynamic**.

Static properties

Static system properties are configured in the file NETNAMES.INI. They become effective on power up and cannot be changed during runtime.

Static properties include:

- Assignment of bus nodes - bus system
- Combination of different bus systems (OPI, MPI)
- Assignment of HMI - NCU (which PCUs/HT6s can monitor which NCUs)
- MCP switchover
- Suppression priorities at switchover (see below)
- Utilization properties:
  - Control unit is alarm/data management **server**
  - Control unit is main or secondary control panel

Dynamic properties

The dynamic properties can be changed during runtime.

The states:

<table>
<tr><th colspan="2">Online</th><th>Offline</th></tr>
<tr><td colspan="2">Normal HMI operating mode with communication between PCU/HT6 and NCU: Operation and/or monitoring possible.</td><td rowspan="3">No communication between PCU/HT6 and NCU:
Operation and monitoring not possible.</td></tr>
<tr><th>Active</th><th>Passive</th></tr>
<tr><td>The operator can operate and monitor.</td><td>Operator cannot operate. He sees a window with header and alarm line and a message indicating the "passive" state.</td></tr>
</table>

Control unit switchover is **enabled**.	Control unit switchover is **disabled**.

Operating the M:N function

The M:N function is operated via the "Channel menu" option.

The channel menu is selected using the "Channel switchover" softkey.

Use the horizontal softkeys to select a channel group (HMI Embedded/HT6: max. 8, HMI Advanced: max. 24 channel groups). Up to eight connections to channels in different NCUs can be set up in one channel group.

The "Channel menu" display shows all the current connections and the associated symbol names.

Suppression strategy

If two PCUs/HT6s are online on one NCU, and a third PCU/HT6 would like to go online, then the latter can "suppress" one of the other two. Communication is then interrupted between this MMC and the NCU.

The algorithm responsible for this suppression is driven by priorities configured in the file NETNAMES.INI.

2.1.2.5 Configurability

NETNAMES.INI

When the M:N system powers up, it must be aware of the existing control units, NCUs and communications links and their properties.

All this information is contained in the configuration file NETNAMES.INI, which is configured before power up.

This present description is mainly intended to provide the necessary knowledge for correctly setting up this configuration file for the M:N concept.

This means that:

- The hardware configuration is displayed
- The properties of the components are defined
- The desired switchovers/assignments are possible

2.1.3 Several operator panel fronts and NCUs, standard functionality

2.1.3.1 System features

General information

The following applies to all M:N applications in which the "control unit management" option is not implemented.

Note

This section does not apply to the HT6, since only one HT6 can ever be operated on an NCU without control unit management.

System features

- The connection between the PCU and the NCU is implemented via:
  - MPI bus (187.5 Kbaud)
  - OPI (1.5 Mbaud):
- The following configurations are possible:
  - "One operator panel front and up to three NCUs"
  - "One operator panel front and up to four NCUs"
  - "Several operator panel fronts and NCUs"
    Connection of operator panels via the OPI interface (X101 on NCU) and the MPI interface (X122 on NCU).
- One of the panel fronts must be an OP030.
- Two MCPs and one HHU can be connected to the MPI or OPI on one NCU.
- The necessary configuration in the NC for the connection of MCPs/HHUs is defined using the basic PLC program (see Function Description, P3: Basic PLC Program).
- Addresses must be specified in the case of data exchange between PLCs via Profibus DP or for global data (double addressing) with PLC-CPU 315 and higher.
  The following applies: NC address = PLC address + 1.
- The maximum number of bus nodes is 32.

2.1.3.2 Functions

Switchover of link to another NCU with the softkey labeled "Connections"

A menu appears in which you can select the connections conn_1, ... conn_n (declared in NETNAMES.INI) via softkeys.
The name (name=...) allocated to the connection in NETNAMES.INI is displayed on the softkeys.

A connection to the new NCU is confirmed by pressing a softkey.

Switchover behavior on OP030

It is not possible to switch over to another bus node online. The connection contained in NETNAMES.INI is permanently configured.

HMI Embedded switchover behavior

The "Connections" softkey is only displayed if more than one link is configured in NETNAMES.INI. When changing to the new NCU, the existing connection to another NCU is interrupted.

HMI applications must no longer need a connection to the previous NCU at the time of link switchover (e.g. for active data backup via V.24 interface). Otherwise the control will issue a message if the connection is required.

HMI behavior with respect to the NCU to which it is switching over is the same way as after a restart, i.e. it is in the operating area which is defined as the start operating area.

HMI Advanced switchover behavior

The "Connections" softkey is only displayed if the "M:N" function is activated on the control. The "M:N" function is activated in the "Commissioning/HMI/Operator panel" menu. Connections are maintained during the switchover process, and the applications which are using these connections also remain active. After the switchover, the HMI is in the same operating area with respect to the new NCU as it was previously with respect to the other NCU.

Possible faults

The NCU with which the connection is to be set up can reject the connection setup. Reason: NCU faulty or the NCU cannot operate any additional control units at this time.

Machine data MD10134 $MN_MM_NUM_MMC_UNITS (number of possible simultaneous HMI communications partners) contains the setting which defines how many control units can be processed by an NCU at one time.

The OP030 uses one unit. A PCU, as supplied, uses two units. Other units (up to 12) are required for larger OEM packages.

Alarms, messages

HMI Embedded, OP030	HMI Advanced
It is only possible to output the alarms of the NCU to which a connection is currently active.	The alarms and messages of all connected NCUs can be processed simultaneously.

Alarm text management

HMI Embedded, OP030	HMI Advanced
Only one version of the alarm texts can be stored on the operator component. The standard alarm texts are stored once in the same formulation for all NCUs. The possible alarms for all connected NCUs must be stored in the area for user alarms.	NCU-specific user alarm texts cannot be created.

HMI connection check

The address of a connected NCU (only on the OPI bus) can be changed in the "Connections/Service" menu. The new address of the NCU is stored on the NCU.

The softkey labeled "Service" is only displayed if the password for the "Service" protection level has been entered. "

When the function for changing the address is started up, a direct connection between the HMI and the relevant NCU must always be set up so as to ensure that the address is not programmed more than once on the bus.

Note

When the NCU is replaced (service case) or if the backup battery fails, the stored address is lost.

The address of the NCU is not lost after a general reset of the NCU.

The address can only be changed via the HMI software.

The display of the current connection in the basic display should be based on the assignment of a unique channel name in machine data:
MD20000 $MC_CHAN_NAME (channel name)

M:N function

The M:N function is operated via the "Control unit management" option.

Prerequisite: Configuration via the NETNAMES.INI file

References:
/IAD/ 840D Commissioning Manual

The channel menu is selected using the "Channel switchover" softkey. Use the horizontal softkeys to select a channel group (HMI Embedded: max. 8, HMI Advanced: max. 24 channel groups); up to eight connections to channels in different NCUs can be set up in one channel group. The "Channel menu" display shows all the current connections and the associated symbol names.

Note

If errors occur during power up (if, for example, connection setup fails), see Section "Power up".

2.1.3.3 Configurability

2 control units: 1 NCU

The diagram below illustrates the connection between two control units and one NCU. In this case, there is a fixed assignment between the MCP and the NCU.

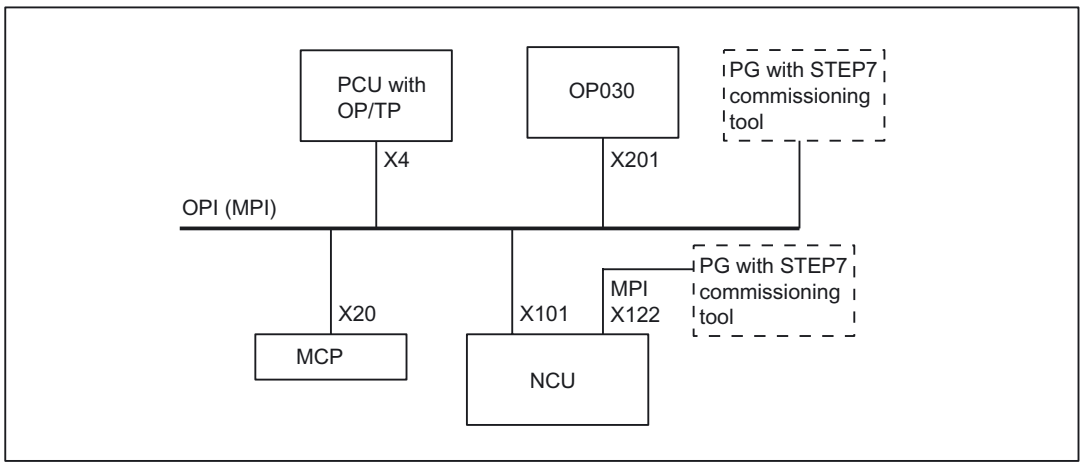


Figure 2-4 Example configuration (M:N in the ratio 2:1)

The control units, NCU and machine control panel are all either connected to the OPI bus or the MPI bus. A **homogenous** network must be provided with respect to these components.

The illustrated configuration enables a large machine tool to be equipped with a control unit on both the front and rear sides, for example.

Features

When operating two control units in the configuration illustrated above, the user will observe the following system operating characteristics:

- For the NCU, there is no difference between inputs from the various control units.
- The control units are mutually independent in terms of data display, i.e. the display selected on one panel is not affected by the display on the other.
- Spontaneous events, such as alarms, are displayed on both control units.
- The protection level set on one control unit will also apply to the second.
- The system does not provide for any further coordination between the control units.

If the user applies the standard configuration shown in the diagram, then no further special settings are required.

1 operator panel: 3 NCUs

One operator panel can be connected to up to three NCUs (see diagram below). In this case, the MCP has a fixed assignment to the relevant NCU.

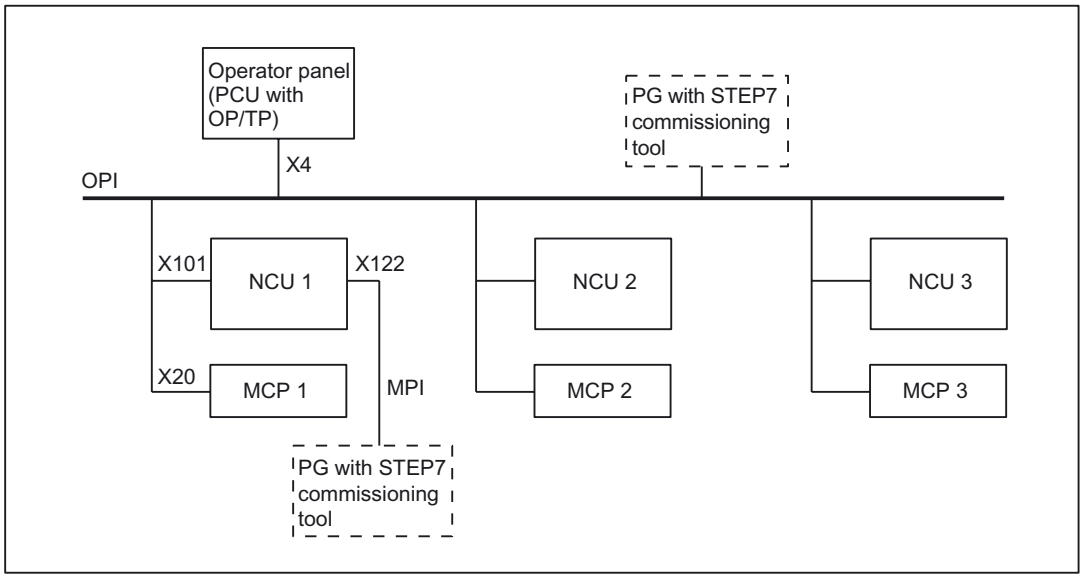


Figure 2-5 Example configuration (M:N in the ratio 1:3)

It is possible to operate several NCUs from one operator panel (several autonomous machines or one large machine with several NCUs). At any given time, only one preselected NCU is connected to the operator panel for operations.

- HMI Embedded also has only one connection for alarms.
- HMI Advanced: The PCU remains connected to all NCUs for alarms.

Features

The operating characteristics are as follows when several NCUs are linked to one operator panel:

- NCU operation:

  The user must select the NCU to be operated by means of a softkey.

  The operator display in the "Connection" operating area displays the name of the connection and of the NCU to which the operator panel is currently linked.

- HMI Embedded:
  - No application should be active on the connection which is interrupted by the changeover to another NCU (Example: data backup via V24). System message "V24 active" is output if an attempt is made to switch over the link when an application is active.
  - The HMI is in the default Start operating area for the new connection (as after HMI restart).
- HMI Advanced:

  When a link to another NCU is set up, the last operating area selected on the previous NCU is immediately available for the new NCU.

OEM solution

As an OEM solution, a PCU with HMI Advanced can be connected via an OPI to up to three NCUs as a program and alarm server (m=1, n=3).

A PG with a commissioning tool can also be connected.

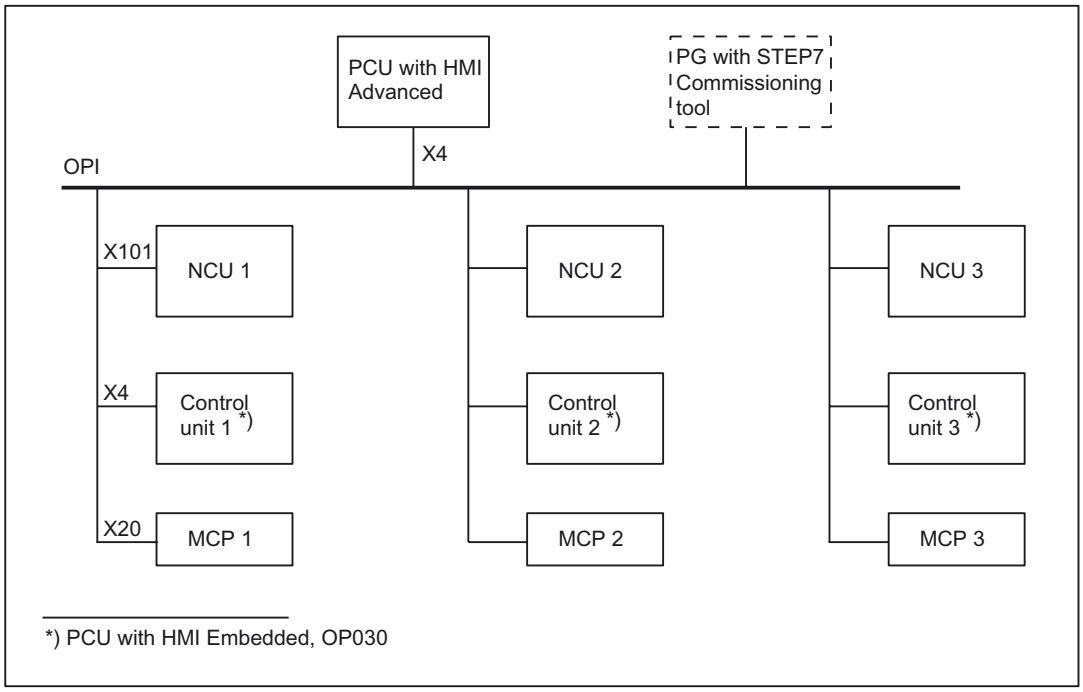


Figure 2-6 Example configuration for OEM solution

Features

The following operating characteristics are typical of the OEM solution illustrated in the diagram above:

- NCU operation:

  The user must select the NCU to be operated by means of a softkey.

  The operator display shows the name of the connection and of the NCU to which the control unit is currently linked.

- HMI Embedded

  Can only be connected to a local NCU.

- HMI Advanced

  When a link to another NCU is set up, the last operating area selected on the previous NCU is immediately available for the new NCU.

1 operator panel: 4 NCUs

In addition to the options described above, it is also possible to create a link between an operator panel with HMI Advanced and up to four NCUs, as illustrated in the diagram below. The MCP and the local operator panel with HMI Embedded have fixed assignments to the relevant NCU in this case.

A second control unit can be connected to the OPI.

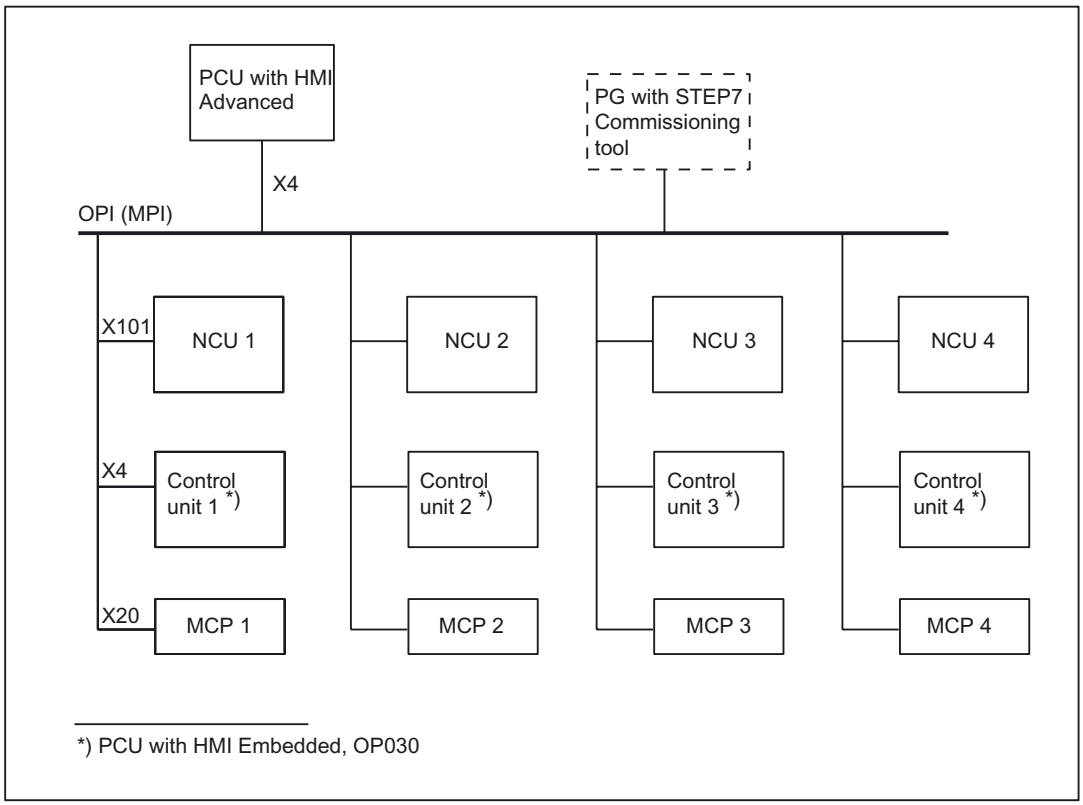


Figure 2-7 Example configuration (M:N in the ratio 1:4)

At any given time, only one preselected NCU can be connected to the HMI Advanced operator panel for operations:

- HMI Embedded also only has one connection for alarms.
- HMI Advanced: The PCU remains connected to all NCUs for alarms.

References

/BH/ Operator Components Manual

/IAD/ Commissioning Manual

/FB1/ Function Manual, Basic Functions, Basic PLC Program (P3)

/FB2/ Function Manual, Extended Functions; Memory Configuration (S7)

The following are described here:

- MPI/OPI bus structure, bus addresses (IAD)
- Bus terminator (IAD, FB2)
- Connection of MCPs via basic PLC program (FB1)
- DIP FIX settings of MCP (IAD)

2.1.3.4 MPI/OPI network rules

Network installations

Please take the following basic rules into account when undertaking network installations:

- The bus line must be terminated at **both ends**. To do this, you switch on the terminating resistor in the MPI connector of the first and last node, and switch off any other terminators.

  Note

  Only two inserted terminating resistors are permitted.

  In the case of HHU/HT6, bus terminating resistors are **permanently** installed in the device.

- It is necessary to apply a 5 V voltage to at least 1 terminator.
  This is automatically supplied as soon as the MPI connector with the terminating resistor fitted is connected to an activated device.
- Drop cables (feeder cable from bus segment to node) should be as short as possible.

  Note

  Any spur lines that are not assigned should be removed if possible.

- Each MPI node must first be connected and then activated.
  When disconnecting an MPI node, first deactivate the connection, then remove the connector.

- Either one HHU and one HT6, or two HHUs, or two HT6s can be connected to each bus segment. **No** bus terminators must be inserted in the distributor boxes of an HHU or HT6.

  If more than one HHU/HPU are connected to a bus segment, this can be done with an intermediate repeater.

- The following cable lengths for MPI or OPI for standard use without repeater may not be exceeded:

  MPI (187.5 kbaud): Max. cable length 1000 m in total

  OPI (1.5 Mbaud): Max. cable length 200 m in total

Note

Piggy-back connectors are not recommended for power connections.

For more information about bus communication, see:
References:
/IAD/ 840D Commissioning Manual

2.1.4 NCU link

2.1.4.1 General information

Use

The NCU link, the link between several NCU units of an installation, is used in distributed system configurations.

When there is a high demand for axes and channels, e.g. in case of revolving machines and multi-spindle machines, the computing capacity, configuration options and memory areas can reach their practical limits when only one NCU is used. Several NCUs interconnected with an NCU link module provide a scalable solution which fully meets the requirements of machine tools of this type. The NCU-Link module offers fast NCU-NCU communication based on a synchronized 12 Mbaud PROFIBUS interface.

Note

NCU-Link is available in case of SINUMERIK 840D with link module and HMI Advanced.

⚠ WARNING

If Safety Integrated (SI) is used in machines with NCU Link, then it is possible, when:

- the configuration of the system-integrated SI functions does not include the link, containerlink and/or lead-link axes.
- in the configuration of the safety-oriented inputs/outputs (SGE/SGA) no safety-oriented signal exchange is necessary between the controls involved, or the signal exchange between the controls involved takes place exclusively via the safety-oriented inputs/outputs (SGE/SGA) in accordance with the functional description of Safety Integrated (e.g. consideration of the 3-terminal concept).

However, if axial system-integrated SI functions of link or container-link axes are still used, then this is done under the responsibility of the machine manufacturer. Hence there are boundary conditions for the SI functions Stop D/E, SG, SE, SN, as well as for commissioning, diagnosis and for the SI acceptance test.

Support can be provided by the Center of Competence Service (CoCS) - SINUMERIK Safety Integrated. Please contact the telephone numbers and addresses given in the foreword under Technical Support.

2.1.4.2 Types of distributed machines

Machine characteristics

Rotary indexing machines/multi-spindle machines exhibit the following characteristics:

- Global, cross-station units (not assignable to one station):
  - Drum/rotary switching axis and
  - units which change from station (position) to station (position):

    In the case of rotary indexing machines: → the rotary axis for workpiece clamping with multi-face machining

    In the case of multi-spindle machines:
    → spindle, quill
- Station-related (position-related), fixed-location units:
  - Slides, milling/drilling units used on the part that is changed from station to station for a machining task.

Applications

Rotary indexing machines (**RVM**) and multi-spindle machines (**MS**) are used as highly productive machines in medium and large batch production.
Their main advantage is that many machining steps can be performed on the workpiece in one clamp.

NCU assignments

The numerous axes on these machines are assigned to different NCUs based on the RVM/MS configuration (e.g. one NCU for each machining unit or group of machining units). The global units are assigned to a separate NCU or distributed accordingly.

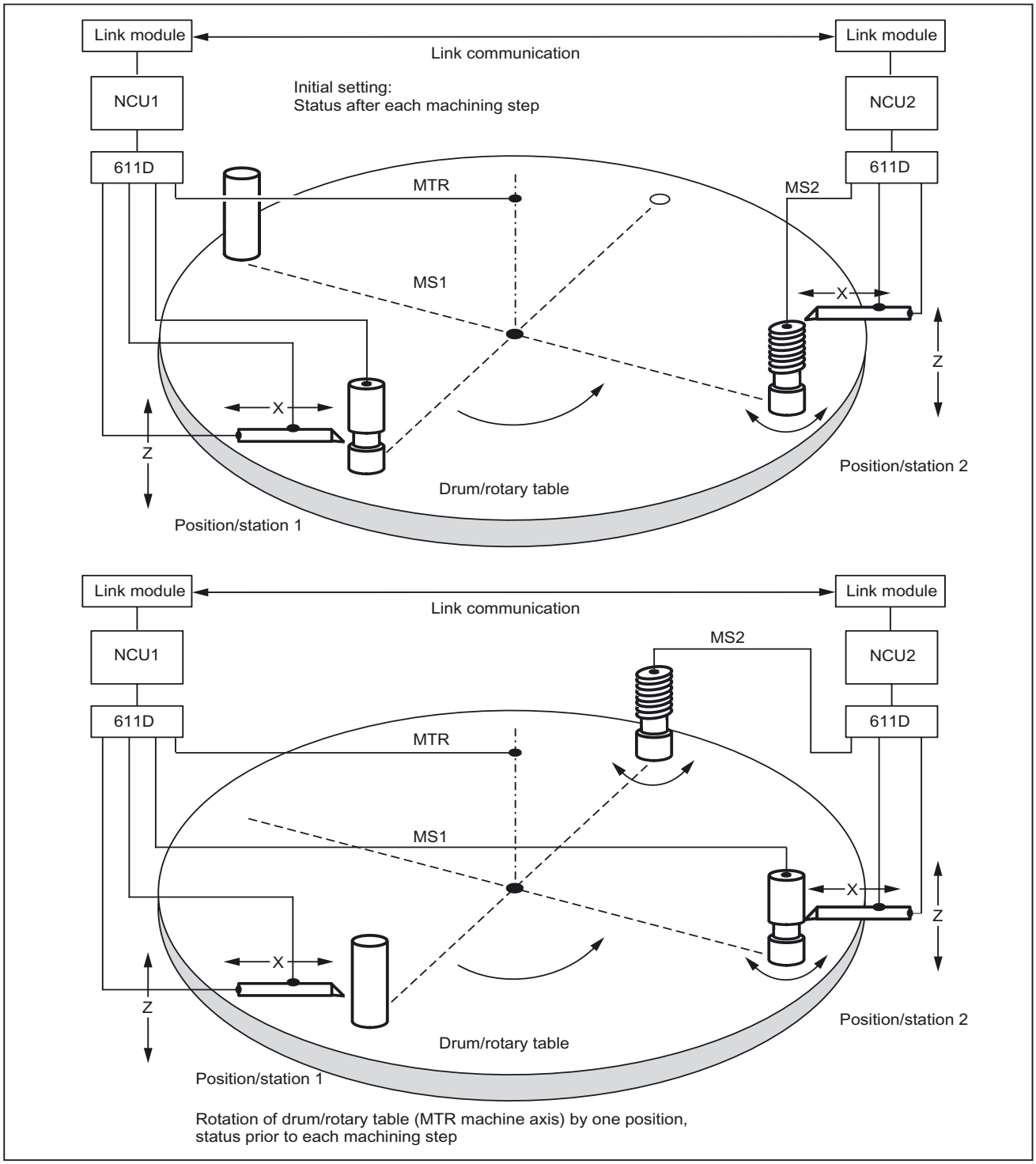


Figure 2-8 Sectional diagram of a drum changeover

When advancing the rotary table with RVM or the drum with MS, the axis holding the workpiece moves to the next machining unit.

The axis holding the workpiece is now assigned to the channel of the machining unit. This is on another NCU in the example, but this is not necessarily the case.

As the above diagram shows, machine axis MS1, which is physically controlled by NCU1, is brought into position/station2 through rotation of the drum/rotary table. To ensure that a coordinated machining operation between the slide and spindle can now take place in position/station2, the commands for spindle MS1 are transferred in this position by means of link communication. Spindles MS1, MS2, ... are **link axes**.

Physical axes can only be subordinate to the motion control of one NCU channel at any one time. However, the motion control initiative for an axis can be assigned to different NCU channels in succession.

Solutions

The **Link Axis** property has been introduced so as to make a physical axis available to several different NCUs. In the diagram above, MS1 becomes the link axis from the point of view of NCU2 (bottom diagram) after it has been turned to position/station2.

The configuration concept **axis container** is available for variable assignments of channel axes to machine axes according to axis group.

All link axes which are moved to the next position/station by a particular drum/rotary table must be managed in the same axis container.

2.1.4.3 Link axes

Link axes

- Definition:

  A link axis is an axis whose drive control and position control are subordinate to another NCU or which is the local axis of the NCU concerned, but which can be addressed by another NCU.

- The software option "Link axis" must be installed.

Coordination

- The alternate use of a physical axis by several NCUs is dependent on all the relevant NCUs being aware of the status and data of the particular axis and on their ability to coordinate use of the axis.

Interpolation

- Local axes and link axes can be interpolated together through motion control by means of one NCU.

Hardware

- The NCUs involved in alternate use of axes across NCU limits must be equipped with a **link module**. The NCU link module offers fast NCU-to-NCU communication based on a synchronized 12-Mbaud Profibus interface.

References:
/PHD/ 840D NCU Configuration Manual

The following description provides the information required to configure, program and coordinate the distributed machines shown in the drawing.

2.1.4.4 Flexible configuration

Axis container

On rotary indexing machines/multi-spindle machines, the axes holding the workpiece move from one machining unit to the next. Since the machining units are subject to different NCU channels, the axes holding the workpiece must be dynamically reassigned to the corresponding NCU channel if there is a change in station/position. **Axis containers** are provided for this purpose.

Only one workpiece clamping axis/spindle is active at any one time on the local machining unit. The axis container provides the possible connections to all clamping axes/spindles. Only **one** of these is ever **activated** for the machining unit.

The following can be assigned via the axis container:

- Local axes and/or
- Link axes

Switching between the available axes defined in an axis container works by cyclical shifting (rotation) of the entries in the axis container.

The shift can be triggered by the **part program**.

Range of validity

Axis containers with link axes are cross-NCU (NCU-global) devices which are coordinated via the control.

It is also possible to have axis containers that are only used for managing local axes.

2.1.4.5 User communication across the NCUs

Link variables

- Every NCU connected by means of a link module can address uniformly accessible **global link variables** for all connected NCUs. Link variables can be programmed in the same way as **system variables**. As a rule, the machine manufacturer defines and documents the meanings of these variables.
- Applications for link variables:
  - global machine states,
  - Workpiece clamping open/closed
  - ...
- Data volume comparatively small
- Transmission rate very high because information relevant to main run is available for exchange.
- These system variables can be accessed from the **part program** and from **synchronized actions**. You can configure the size of the memory area for global system variables.
- All connected NCUs require one interpolation cycle before they can consistently read a new value in a global system variable.

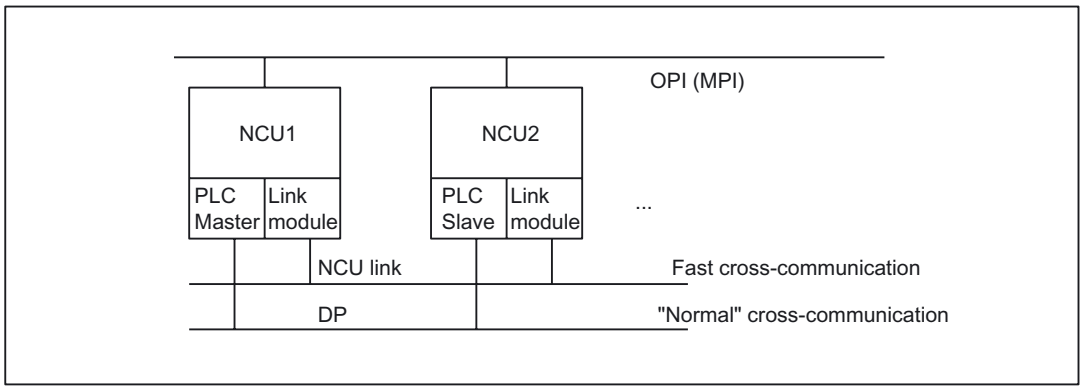


Figure 2-9 Communication overview

Note

On installations without an NCU link, the link variables can also be used NCU-locally as an additional means of cross-channel communication.
In this case no interpolation cycle elapses between writing and reading.

References:
/FBSY/ Function Manual, Synchronized Actions

2.1.4.6 Lead link axes

Following axis movements

The configuration illustrated below shows how to traverse following axes on several NCUs (NCU2 to NCU n in the diagram) in relation to the movement of the leading axis controlled by another NCU (NCU 1 in the example).

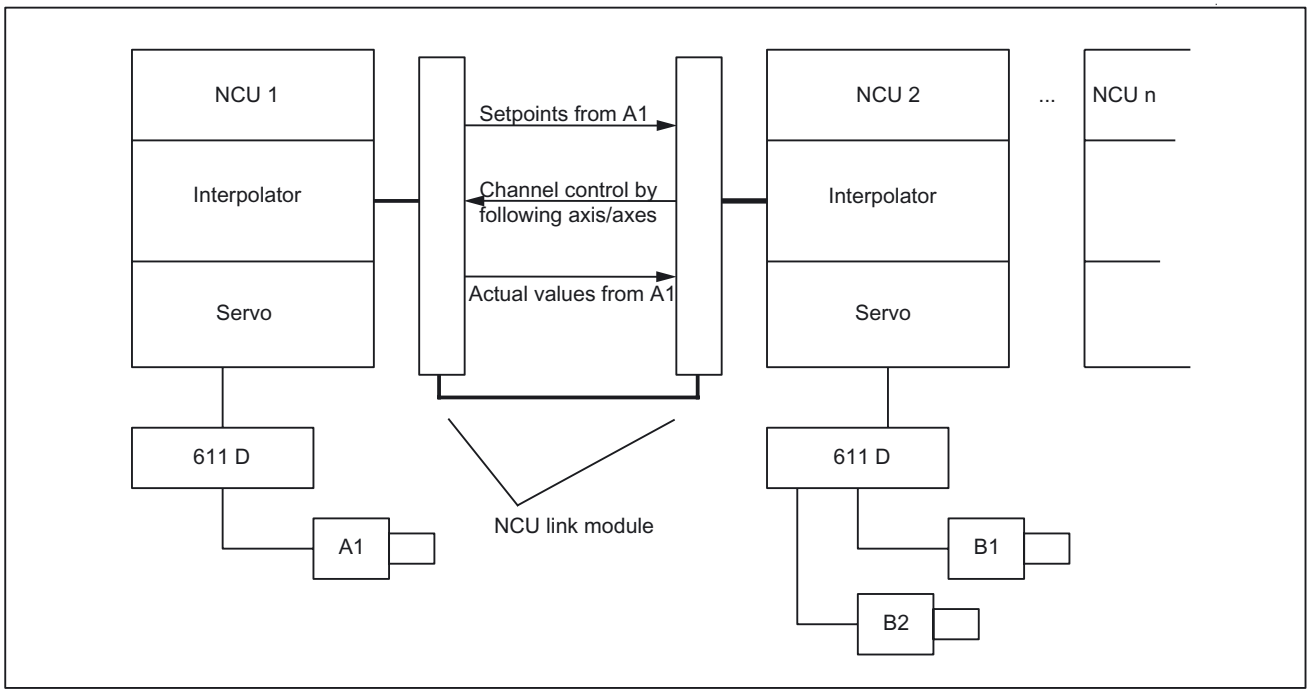


Figure 2-10 NCU 2 moves the following axes via a coupling to a leading axis on NCU1

If the supplementary conditions described in the "Detailed description" are complied with, it is possible for several NCUs in the whole NCU link group to traverse leading axes and for other NCUs to derive following axis movements from them. The leading axes must only be traversed by the NCUs to which they are physically connected (local NCUs). (Home NCUs).

2.1.4.7 NCU link with different interpolation cycles

Function

An extension of the link concept whereby NCUs are connected to link modules with different interpolation cycle settings offers additional application possibilities. This functionality is also called "Fast IPO link", because when different cycles are set, one of the connected NCUs has the fastest interpolation cycle.

Application guideline

Whereas the NCU/NCUs with a standard interpolation cycle operate(s) axes and spindles with standard requirements with respect to dynamic performance and accuracy, the NCU/NCUs with a faster interpolation cycle operate(s) one or a small number of axes with higher requirements with respect to dynamic performance and accuracy.

Examples:

Eccentric turning (cams, pistons, etc.)

- C and Z axes: standard requirements
- X axis: high requirements with respect to dynamic performance and accuracy

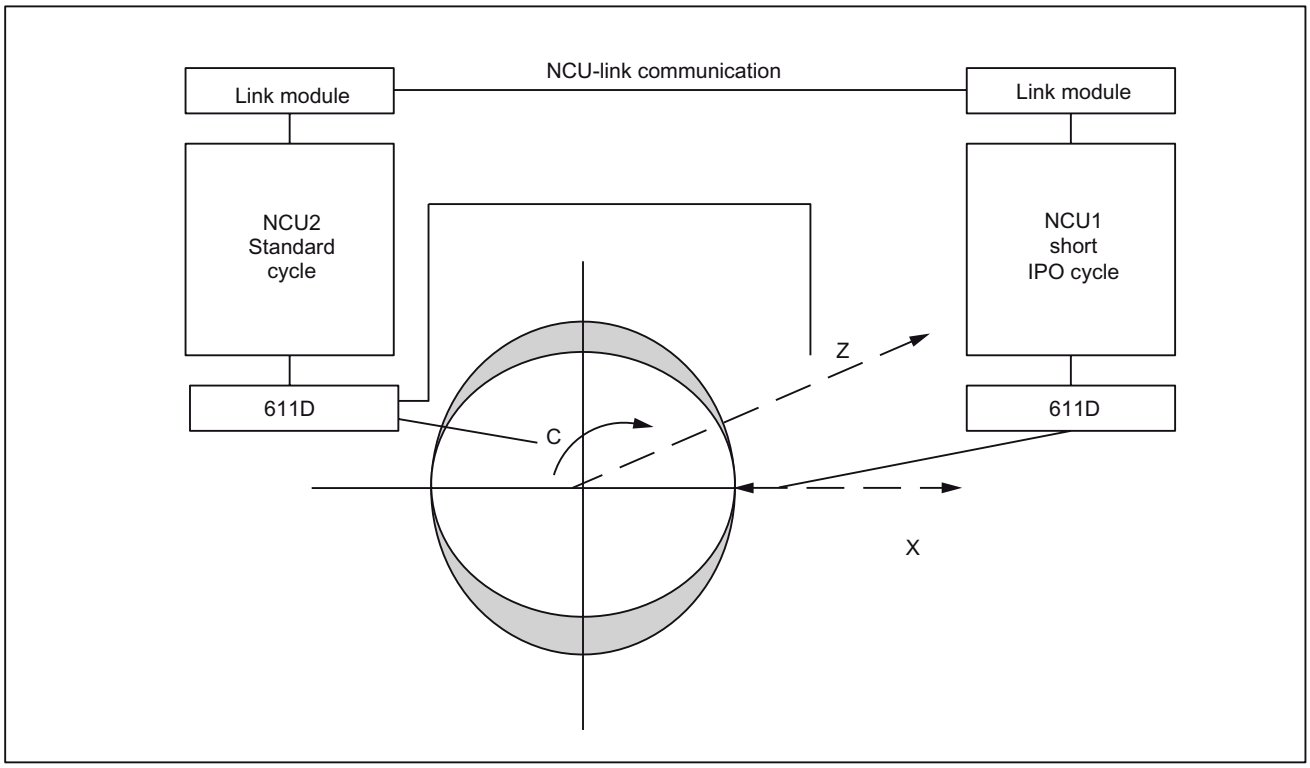


Figure 2-11 NCU link with different interpolation cycles

Essential features

- Cross-NCU interpolation of fast (X) and standard (C,Z) axes/spindles (see diagram).
- The part program is running on the NCU with the faster interpolation cycle and can "see" the other axes as link axes or container link axes.
- Communication between the NCUs takes place via the link modules in the cycle of the slower NCU/NCUs (link cycle).
- The slower interpolation cycle is an integer multiple of the faster interpolation cycle.
- Due to the different cycle clocks used, it is necessary to comply with the supplementary conditions described in the "Detailed description".
- Contour accuracy between the faster and slower interpolating axes is ensured provided the slower interpolating axes are not accelerated during the machining operation.

2.2 Several operator panel fronts and NCUs with control unit management option

The following section provides a detailed description of the preparations and implementation of the operating steps for the M:N concept.

Proceed as follows

1. Defining the configuration:
   - On the hardware side: by interconnecting components via bus systems
   - On the software side: Configuration of the static properties with the help of configuration file NETNAMES.INI (see below). These static properties become effective on power up and cannot be changed during runtime.
2. Control unit switchover function in the PLCs of the relevant NCUs. Control unit switchover is included with the Toolbox and comprises several blocks. They perform the following tasks:
   - Checking of switchover conditions
   - Priority-based suppression
   - Switchover
3. Dynamic properties (such as online/offline states) can be changed during runtime within the limits specified by the NETNAMES.INI file.

2.2.1 Hardware structure

As described in Section "Short Description", a complex system can consist of M control units and N NCUs.

The hardware components are connected to one another via the bus (MPI and/or OPI). The relationships between the bus nodes (identification, properties, assignment and switchover) are software-controlled.

2.2.2 Properties

Client identification

The assignment between bus nodes and the bus system is static and cannot be changed during runtime. It is configured once in the file NETNAMES.INI.

The client identification (CLIENT_IDENT) which the control unit uses for logging on to an NCU in order to set up an online connection is composed of the bus type and bus address.

Properties

The M:N system features control units with the following properties:

Server		Control panel	
Maintains a constant 1:N connection		Can be switched over to different NCUs and maintains a constant 1:1 connection (only one at any one time!). The operator can operate and monitor. Connection is set up when HMI goes online and cleared when it goes offline.	
Alarm server (HMI Advanced)	Data management server (HMI Advanced)	Main control panel	Secondary control panel
Receives the alarms from all NCUs in an M:N system. From its side, a constant 1:N connection is maintained. The process "Receive alarms" is always active and runs in the background.	Sets up all the connections configured for it in NETNAMES.INI on power up and maintains a constant 1:N connection. Can receive, manage and distribute data based on the job list concept.	Example: Main control panel for rotary indexing machines can be connected to all machining stations.	Example: Secondary control panel for rotary indexing machines can only be connected to one of two adjacent machining stations.
Cannot be suppressed (see Section "Suppression")	Cannot be suppressed.	Cannot be suppressed.	Can be suppressed by the main or secondary control panel.

Distribution of properties among the HMI types:

Property	HMI Advanced	HMI Embedded/HT6
Server	x	
Main control panel	x	x
Secondary control panel	x	x

HMI is both server and main control panel at the same time

As a server, the HMI maintains constant 1:N connections; as a main control panel it has a switchable 1:1 connection.

If the HMI is switched over to another NCU as a control panel, it occupies the same connection which it already has as a server. A new connection is not set up.

Permissible combinations in one installation

If a server (alarm/data management server) is configured in an M:N system, it also acts as a main control panel.

Only one control unit in an M:N system can have the following properties:

- Windows-HMI (HMI Advanced): Server and main control panel

or

- Non-Windows-HMI (HMI Embedded/HT6): Main control panel

Any number of secondary control panels is possible.

Note

If the function **Execution from external source** is to be available, **one** operator panel in the system must be designated as a **server**.

2.2.3 Configuration file NETNAMES.INI

As the hardware components can be freely combined (see Section "Hardware structure"), it is necessary to provide the system with information about the components which are connected, how they are connected to each other, and how they interact.

In particular, it is necessary to regulate the competition among the different control units for the limited number of available interfaces (suppression, see Section "Suppression").

For this purpose each PCU/HT6 has a configuration file NETNAMES.INI in which the the configuration parameters must be stored.

2.2.4 Structure of the configuration file

The structure of the configuration file NETNAMES.INI is as follows:

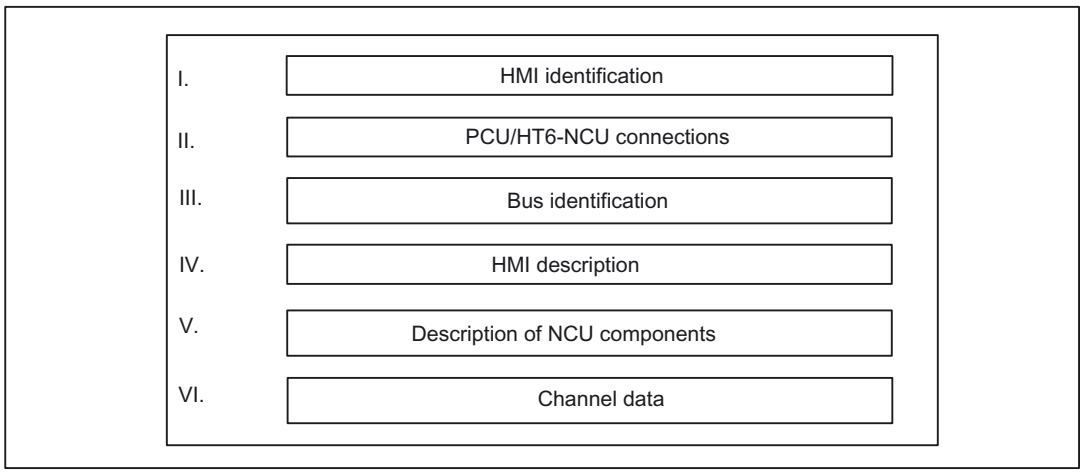


Figure 2-12 Structure of the configuration file NETNAMES.INI

In the following tables,

- Parameters which the user may need to change are printed in *italics*
- Parameters which can be used as alternatives are specified separated by |.

I. HMI identification

PCU/HT6 identifier to which NETNAMES.INI applies:

Element	Explanation	Example
[own]	Header	[own]
owner = *Identifier*	HMI identification	owner = MMC_1

II. PCU/HT6 NCU connections

Configuring the connections between PCU/HT6 and the NCUs:

Element	Explanation	Example
[conn *Identifier*]	Header	[conn MMC_1]
conn_i = *NCU_ID*	Configuring the NCU connection(s) i = 1, ..., 15	conn_1 = NCU_1 conn_2 = NCU_2 ... conn_i = NCU_i

III. Bus identification

Defines which bus the HMI is connected to:

Element	Explanation	Example
[param network]	Header	[param network]
bus = *OPI* | *MPI*	Bus designation	bus = OPI

opi: Operator panel front interface with 1.5 Mbaud

mpi: Multi-point interface with 187.5 Kbaud

Note

The baud rate is automatically detected on the HMI Embedded/HT6.

IV. HMI Description

Characteristics of the control unit:

Element	Explanation	Example
[param *Identifier*]	Header	[param MMC_1]
mmc_typ = *Type/connection identifier*	HMI characteristics (see below)	mmc_typ = 0x40 HMI is both server and main control panel See below for explanations
mmc_bustyp = *OPI* | *MPI*	Bus the HMI is connected to	mmc_bustyp = OPI
mmc_address = *Address*	HMI address	mmc_address = 2
mstt_address or mcp_address = *Address*	Address of MCP to be simultaneously switched over. If none is implemented, there is no MCP to be switched over.	mstt_address = 6 or mcp_address = 6
name = *Identifier*	Any name allocated by the user (optional, max. 32 characters)	name = HMI_LINKS
start_mode = *ONLINE* | *OFFLINE*	State after power up. If ONLINE, link is set up via DEFAULT channel entry to the associated NCU. OFFLINE: No link is set up immediately after power up. Important: In addition, it is necessary to set the entry NcddeDefaultMachineName = local in the [GLOBAL] section of mmc.ini.	start_mode = ONLINE (During power up HMI is connected online to the NCU to which the channel is assigned via channel data (see VI) DEFAULT_logChanGrp, DEFAULT_ log_Chan).

In the case of HT6, the integrated MCP is always switched over simultaneously. The integrated MCP is responsible for address assignments of HT6. Only values between 1 and 15 are possible for this reason.

Note

Note that the NCU configured via the DEFAULT channel must be the same as the NCU specified under NcddeDefaultMachineName in file **MMC.INI**.

Explanatory notes on **mmc_typ**:

mmc_typ contains type and connection identifiers for the control units and is transferred to the PLC in the event of a switching request. mmc_typ is evaluated as a priority for the suppression strategy. See Section "Suppression".

Bit 7 = --	(reserved)
Bit 6 = TRUE:	HMI is the server (HMI Advanced) and cannot be suppressed.
Bit 5 = TRUE:	Operator panel/HT6 is the main control panel.
Bit 4 = TRUE:	Operator panel/HT6 is the secondary control panel.

Operator panel = PCU (incl. HMI Advanced/Embedded) with OP/TP

The user can specify four additional HMI types which the control unit switchover function of the PLC also takes into account in its suppression strategy:

Bit 3 = TRUE:	OEM_MMC_3
Bit 2 = TRUE:	OEM_MMC_2
Bit 1 = TRUE:	OEM_MMC_1
Bit 0 = TRUE:	OEM_MMC_0

If no mmc_typ is entered in file NETNAMES.INI, then the HMI/HT6 powers up by the method defined for standard functionality.

V. Description of NCU component(s)

A separate entry must be generated for every single NCU component connected to the bus.

Element	Explanation	Example
[param *NCU_ID*]	Header	[param NCU_1]
name= *any_name*	Any name assigned by the user; is output in the alarm line (optional, max. 32 characters)	name= NCU1
type= *NCU_561*/*NCU_571*/ *NCU_572*/*NCU_573*	NCU type	type= NCU_572
nck_address = *j*	Address of NCU component on the bus: *j* = 1, 2, ..., 30 *)	nck_address = 14
plc_address = *p*	Address of PLC component on the bus: *p* = 1, 2, ..., 30 *) (Only necessary for MPI bus; in the case of the OPI bus: j = p)	plc_address = 14

*) For MPI bus: Since the associated NCU always occupies the next-higher address than the PLC, the PLC address must not be 31. Address 31 can be assigned to a PCU, for example.

Note

If the bus node addresses on the MPI bus are configured in conformance with SIMATIC, the configuring engineer can read out the assigned addresses using a SIMATIC programming device and use them to create the NETNAMES.INI file.

VI. Channel data

The control unit switchover option can work only if the control unit knows how channels are assigned to NCUs so that it can set up links between the control unit and NCUs. (Channel menu).

Concept

The following steps are necessary:

1. Definition of technological channel groups
2. Assignment of channels to groups
3. Assignment of NCUs to channels
4. Definition of power-up link

NCUs are addressed indirectly on the control unit on the basis of channel group and channel. See Section "Operator interface".

References:

/IAM/ Commissioning Manual HMI

/FB1/ Function Manual, Basic Functions; "PLC Basic Program"

Element	Explanation	Example
[chan *identifier*]	Header (channel menu of MMC_1)	[chan MMC_1]
DEFAULT_logChanGrp = *group*	Channel group of channel during power up (4.)	DEFAULT_logChanGrp = Mill1
DEFAULT_logChan = *channel*	Selected channel during power up (4.)	DEFAULT_logChan = channel11
ShowChanMenu = *TRUE* / *FALSE*	TRUE Display channel menu	ShowChanMenu = TRUE
logChanSetList = *group list*	List of channel groups (1.)	logChanSet = mill1, mill2
[*group*]	Header (2.)	[mill1]
logChanList = *channel1, channel2,...*	Groups channels separated by comma (2.)	logChanList = channel11, channel12, channel13
[*channel*]	Header (3.)	[channel11]
logNCName = *identifier*	Log. identifier of an NCU (3.)	logNCName = NCU_1

Element	Explanation	Example
ChanNum = i (i = 1, 2, 3,...)	Number of channel configured for associated NCU (3.)	ChanNum = 1
And so on for all channels in group		
Continue with next group and its channels		

A complete example of how to configure the channel menu can be found in "Configuration file NETNAMES.INI with the control unit management option".

2.2.5 Creating and using the configuration file

Syntax

The configuration file must be generated as an ASCII file. The syntax is the same as that used in Windows *.ini files.

In particular, the following is applicable:

- Passwords must be typed in small letters.
- Comments can be inserted in the parameter file (limited on the left by ";" and on the right by end of line).
- Blanks may be used as separators at any position except in identifiers and passwords.

HMI Embedded, OP030, HT6

The NETNAMES.INI file created on a PC or programming device is loaded, as described in

References: FBO/IK/ Configuration Operator Interface OP 030/Installation Kit

via the RS 232 interface and permanently stored in the FLASH memory of the control units.

HMI Advanced

The NETNAMES.INI file can be edited directly with an editor (in menu "Commissioning/HMI/Editor" or DOS_SHELL) on the hard disk of the operator component. The NETNAMES.INI file is stored in the installation directory C:\USER\.

Example

For a sample configuration file, see Section "Examples".

2.2.6 Power up

Defaults standard functionality

The following defaults are applied (standard M:N = 1:1) if no NETNAMES.INI configuration file is loaded into the HMI Embedded/OP030/HT6 or if the file cannot be interpreted:

- The bus type used is detected automatically.
- HMI has address 1.
- OP030 has address 10.
- NCU and PLC both have address 13 for an OPI bus.
- NCU has address 3 and PLC address 2 for an MPI bus.

With option

If, however, a special NETNAMES.INI file is created, then it must correspond exactly to the actual network on account of the special features described below.

If an M:N-capable control unit fails to set up a link to the NCU during power up or if a configuring error occurs, the control unit switches over to OFFLINE mode. In this mode, the operator can switch over to the area application via the Recall key and from there to the commissioning area.

Compatibility

The use of the above defaults guarantees compatibility to all software versions for operator panel operation.

Power up with HMI Embedded/HT6

An HMI Embedded/HT6 control unit can only set up an active link to the NCU if the configuration in NETNAMES.INI complies with the description in Section "Structure of the configuration file". HMI Embedded, HT6 and OP030 can power up in parallel on one NCU, because as bus nodes they have different addresses. The OP030 can be used as a second operator panel front with a fixed assignment to an NCU.

If the configured addresses do not match the real addresses (NC/PLC address), the commissioning engineer can use the following key sequence to power up on an NCU that is not configured.

Sequence

1. HMI/HT6 boots on the NCU with bus address 13 if the NETNAMES.INI file has not been changed (original factory settings).
2. The file NETNAMES.INI has been changed, this message appears:

   "HMI Embedded version xx.xx.xx: waiting for connection ..."

   – Press key "1", this message appears:

   "choice: '1'=set new start-address, '^' =boot"

   – Press key "1", the bus addresses of all nodes connected to the bus are displayed. This message appears:

   "Please try one of the shown addresses or press '^' to reboot

   '1',_,_,_,_,'6',_,..._,'D',_,..."

   – Press keys "D" and INPUT

   – HMI/HT6 boots on the NCU with bus address 13 (if an NCU is configured under the address found).
3. Enter new NC address in the Commissioning/NC/NC address operating area and confirm with "Yes".
4. NC reset (new address only takes effect valid after NC reset)
5. Configure connection/channel menu in the NETNAMES.INI file and transfer to the HMI/HT6.
6. After the NCU addresses have been assigned, the bus can be wired for M:N operation.

Note

You can operate an OP030 and a PCU (HMI Embedded)/HT6 on an interface without assigning parameters (various bus addresses are available in the delivery state).

Power up with HMI Advanced

The HMI Advanced can power up even if the link to the NCU cannot be made due to errors in the configuration.

An NCU address can be specified explicitly through the entry of a "1:1" connection in the "Commissioning/HMI/Operator panel" menu. When the HMI powers up again, the communications link between the HMI and NCU/PLC will work properly.

Sequence

1. HMI boots on the NCU with bus address 13, if the NETNAMES.INI has not been not changed (original factory settings).
2. NCU bus address has been changed, this alarm apppears:

   "120201 name: communication failed"

   – Set the connection to 1:1 in the Commissioning/HMI/Operator panel operating area and enter "13" as the NC address

   – Confirm with OK and boot the HMI
3. Point no. 6. As for HMI Embedded

Note

In the event of an error, check the active bus nodes in the menu:
- Commissioning/NC/NCK addresses (HMI Embedded, HT6 and HMI Advanced)
- Commissioning/HMI/Operator panel (HMI Advanced)

Power up with HMI Advanced option

If the control unit switchover option is installed, a configuring problem can be corrected as follows:

1. Select the channel menu with the input key
2. Go to the area switchover screen by pressing Recall
3. Select commissioning.

Required documentation

References:

/BH/ Operator Components Manual

/IAD/ Commissioning Manual

/FB1/ Function Manual, Basic Functions; P3, Basic PLC Program

The following are described here:

- MPI/OPI bus structure, bus addresses, /IAD/
- Bus terminator, /IAD/, /FB/S7
- Connection of MCPs via basic PLC program, /FB/, P3
- DIP FIX settings of MCP, /IAD/

Note

After performing series machine commissioning, a Power On must be performed on the PCU so that the bus nodes (PLC, NC, PCU) can resynchronize.

2.2.7 HMI switchover

With the M:N concept, you can change the control unit properties and states configured in the NETNAMES.INI file during runtime.

For example, the user can intervene in order to

- switch over control units (see Section "Connection and switchover conditions")
- switch over MCPs (see Section "MCP switchover").

A maximum of two control units can be online at the same time on an NCU. A suppression strategy exists in order to avoid conflict situations (see Section "Suppression").

The HMI properties are configured for each control unit in the NETNAMES.INI file. If a control unit wants to go online on an NCU via the switchover protocol, its parameters are passed on to the PLC of the relevant NCU. The PLC program **Control Unit Switchover** evaluates the parameters:

- Check suppression conditions
- Switchover if necessary

2.2.8 Suppression

A maximum of two control units can be online on an NCU. If this is the case, and another PCU/HT6 wants to go online, it must be ensured that no conflicts occur. This is achieved by means of the suppression algorithm described below.

Sequence

- The PLC sends an offline request to the control unit to be suppressed.
- The control unit returns a positive or negative acknowledgement to the PLC:
  - If the acknowledgement is positive, the control unit is suppressed (see below), it terminates communication with the NCU and goes into offline mode.

    If an MCP is assigned to the PCU, this is deactivated by the PLC.

    The integrated MCP is always assigned on the HT6 and is thus also deactivated.
  - A negative acknowledgement is output if processes that cannot be interrupted are running on the control unit, e.g. operation via RS-232 or data transfer between NCU and PCU. In this case the control unit is not suppressed and remains online.

Suppression strategy

The PLC program "Control Unit Switchover" operates according to the

- priorities of the control units and
- the active processes

The priority depends on the parameter **mmc_typ** in configuration file NETNAMES.INI (see Section "Structure of the configuration file"). The Control Unit Switchover program on the PLC evaluates this parameter according to the following table:

HMI property	Priority
Server	6
Main control panel	5
Secondary control panel	4
OEM-MMC 3	3
OEM-MMC 2	2
OEM-MMC 1	1
OEM-MMC 0	0

Suppression rules

The following rules apply for control unit suppression:

- Higher priority suppresses lower priority.
- In the event of identical priority, the active control unit is suppressed.

The following restrictions apply:

- Servers cannot be suppressed, as they require a permanent connection to each NCU.
- Control units on which the following processes are active cannot be suppressed:
  - Data transfer, e.g. from/to NCU
  - Control unit is in the process of switching over to this NCU
  - Control unit is just changing operating mode
  - OEM disables switchover

2.2.9 Connection and switchover conditions

Proceed as follows to allow a previously offline control unit on a particular NCU to go online or to switch an online control unit over to another NCU:

PCU

1. Press the channel switchover key to call up the channel menu on the PCU.
2. Select the channel group via a horizontal softkey.
3. Select the appropriate vertical softkey for the channel. See "Implementation of control unit switchover".

HT6

1. Activate the "Panel Function" by selecting the key with the same name.
2. Select the "Channel" softkey.
3. Select the channel group.
4. Select the channel.

   If the required channel is not included in this group, then you can return to point no. 2 by pressing the "Recall" key.

The PCU/HT6 is then switched to online operation or to another NCU, provided that its change in status is not blocked by one of the following conditions (displayed in message line).

Messages for PCU switchover	
HMI Embedded	Message Text
109001	No switchover: Switchover disable set in current PLC
109002	No switchover: Target PLC occupied, try again
109003	No switchover: Switchover disable set in target PLC
109004	No switchover: PLC occupied by higher-priority PCUs
109005	No switchover: No PCU can be suppressed on target PLC
109006	No switchover: Selected channel invalid
109007	Channel switchover in progress
109009	Switchover: Error in internal state
109010	Suppression: Error in internal state
109012	Control unit switchover, PLC timeout: 002
109013	Activation rejected

Note

Corresponding messages are output without a message number on HMI Advanced.

Additional messages can be generated in HMI Embedded/HT6 and HMI Advanced indicating the current status or errors in the configuration or operating sequence.

For more information, see

References: /DA/ Diagnostics Manual, Chapter 1

2.2.10 Implementation of control unit switchover

Control unit switchover is an extension of channel switchover.

Channel switchover

Channel configuration allows channels of selected NCUs to be individually grouped and named. HMI switchover to another NCU is implemented as part of channel switchover functionality.

Channel configuration is based on the file NETNAMES.INI. See "Structure of configuration file".

2.2.11 Operator interface

Function

The operator interface allows you to set up a connection between the control unit and one of the connected NCU/PLC units in every operating area.

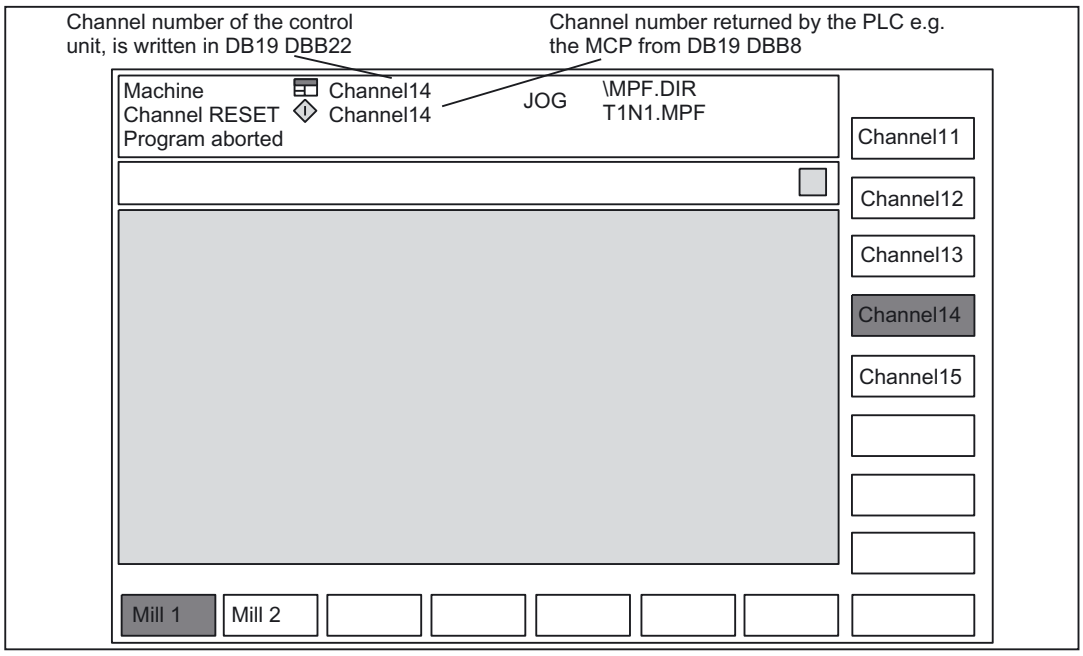


Figure 2-13 Channel menu (the comments refer to the 1st HMI interface)

Only the channels of the relevant group are displayed.

Activate the channel switchover key. The currently existing connection is displayed by means of the highlighted softkeys (horizontal, vertical) if the channel menu is active.

Channel switchover

You can switch over to other channels by means of the vertically arranged softkeys.

Group switchover

You can switch to another group by means of the softkeys on the horizontal menu (see Section "Implementation of control unit switchover"); the channels of the currently selected group are now displayed on the vertical softkeys. Switchover to another channel (and if necessary to another NC) only takes place upon activation of a vertical softkey.

NC switchover

You can change to another NC via the vertical softkeys if the channel is not on the current NC.

Procedure: Configure a channel area NCs (horiz. softkeys 1-8) if applicable and link a channel to the vertical softkeys from each NCU.

Note

The softkeys only offer the connections that are really assigned and whose channels are active in the relevant NC.

HT6

The channel menu on the HT6 is structured in two stages. In the first stage you select the channel group and in the second stage, the channel. For details please refer to the HT6 Operating Manual.

2.2.12 Operating mode switchover

Two PCUs/HT6s can be online at the same time on one NCU. In order to avoid both gaining write access to the same data or file simultaneously, there are two operating modes, i.e.:

- the active and
- the passive operating mode.

Only one of the two PCUs/HT6s can be active; the other is passive.

Interaction takes place according to the following rules:

Active operating mode

- The user requests active operating mode by pressing a key on the operator panel front.

  Active mode has the following characteristics:

  – All operations and operating areas are activated.
  – The operator can operate and monitor.
  – The MCP assigned to the control unit is activated.
  – If data transfer processes (e.g. series machine commissioning, various tool management services, commissioning of drive configuration) are running between the other control unit and the shared NCU, the PCU/HT6 cannot become active immediately.

Passive operating mode

- Passive mode takes effect when the other PCU/HT6 has requested active mode.

  Active mode has the following characteristics:

  – The connection to the NCU is maintained.
  – All operations are deactivated.
  – The operator cannot operate: A window is displayed with a header and an alarm line and a message indicating the "passive" state.
  – The global menu is activated.
  – Any services initiated previously (in active mode) remain active (e.g. operation via RS-232, reloading of part programs, executing the job list, alarms).
  – The MCP assigned to the control unit is deactivated.
  – The application window and softkeys are deactivated.

The active operating mode can be selected by 2 different methods:

- Input key
- Channel switchover key and channel selection.

Rules for operating mode switchover

The following rules apply to changes of operating mode (see also Sections "Suppression", "Suppression strategy"):

- A PCU/HT6 which goes online on an NCU is assigned the active operating mode on this NCU.

  If another PCU/HT6 was previously active on this NCU, it changes to passive mode, provided this is permitted by the PLC.
- If two PCUs/HT6s are online, the operating mode is changed by pressing the key ("Input", ENTER, RETURN) used to select the active operating mode.
- The change from the active to the passive operating mode can be rejected by the PCU/HT6 if the current HMI application cannot be aborted or if it is still in progress. Similarly, active mode cannot be selected on a PCU/HT6 if the other PCU/HT6 currently linked to the NCU cannot be switched to passive mode.
- If an online request is issued by a PCU/HT6
  - and no PCU/HT6 is currently online:

    The PCU/HT6 issuing the request goes online and switches to active mode.

    If an MCP is assigned to the PCU, this is activated by the PLC.
  - and a PCU/HT6 is currently online:

    This PCU/HT6 switches to passive mode and is suppressed.

    The requesting PCU/HT6 goes online.
- If two PCUs are online on one NCU and the previously active PCU/HT6 goes offline, it first switches to passive mode. Then the second PCU/HT6 switches to active mode and the first PCU/HT6 disconnects the link to the NCU.

Note

The HMI type is assessed as a priority for the suppression strategy. See Section "Suppression".

If the active PCU/HT6 cannot be switched to passive mode, then the requesting PCU/HT6 is switched to passive mode.

2.2.13 MCP switchover

An MCP cannot be switched over independently of the PCU it is assigned to. It can be switched over only if

- the PCU switches over and
- the MCP address is defined in the HMI parameter block of NETNAMES.INI (see Section "Structure of configuration file").
- MCP_enable is set in the control unit switchover function on the PLC.

Activating/deactivating the MCP

If an MCP is assigned to the PCU in the NETNAMES.INI file, it is activated/deactivated as part of the operating mode change. The MCP switchover in the PLC is called by the operating mode change as a subfunction.

PCU changes operating mode	MCP is
Active -> passive	Deactivated
Passive -> Active	Activated

2.2.14 PLC program "Control Unit Switchover"

Introduction

Control unit switchover is an important controlling function in the overall M:N concept:

- The PLC control unit switchover checks the priorities of the requests and the states of the components involved and switches over if necessary.
- The NCU sets signals and evaluates signals required in connection with the control unit switchover function.

The control unit switchover function is a software package in the toolbox. It is immediately available with its **standard functionality**, but can be modified for special applications according to individual requirements.

Provision is made for two categories of modification:

1. Simple parameterization of standard functionality
2. More fundamental reconfiguring of the control unit switchover function

Reasons for more fundamental (user-specific) reconfiguring can be as follows:

- Suppression strategy which differs from standard functionality
- Operating mode switchover which differs from standard functionality
- Independent handling of override switch for switchover of control unit
- Existence of a 2nd machine control panel on a PCU.

Program structure

The control unit switchover program consists of:

1. FB101/DB101: Online/offline operating mode switchover
2. FB102/DB102: Active/passive operating mode switchover
3. FC103: Machine control panel switchover

Every program section is implemented in a separate function block (FB) or function (FC). The variables are stored in a separate instance data block (DB) for each FB. The control unit switchover **main program** is stored in **FB101.** FB101 must be called in an organization block (OB) in order to activate the functions. FB102 and FC103 are called repeatedly in FB101.

MCP switchover

The MCP switchover is not mandatory. It can be enabled or disabled via the FB101/DB101 variable: **MSTT_enable**. Assignment in the declaration table:
TRUE: MCP switchover is effective.
FALSE: MCP switchover is not effective, no FC103 call

The MCP assigned to the active PCU is always activated. If the operating mode switchover function is disabled, then the MCP assigned to the last PCU to go online is activated.

Power-up condition:

To prevent the previously selected MCP from being activated again when the NCU is restarted, input parameter **MCP1BusAdr** must be set to 255 (address of 1st MCP) and **MCP1Stop** to TRUE (deactivate 1st MCP) when FB1 is called in OB100.

Enabling commands:

When one MCP is switched over to another, any active feedrate or axis enabling signals may be transferred at the same time.

Note

The override switch of the new MCP is immediately effective following an MCP switchover.

The keys actuated at the time of the switchover continue to take effect. If no MCP is installed on the newly selected PCU, then it will not be possible to cancel the key functions from the PCU. Measures for situations of this type must be implemented in the PLC user program.

Operating mode switchover

The operating mode switchover is not mandatory. It can be enabled or disabled via the FB101/DB101 variable **aktiv_enable**. Assignment in the declaration table:
TRUE: Operating mode switchover is effective
FALSE: Operating mode switchover is not effective, no FB102 call

Error messages

If disturbances occur (e.g. interface signal failure) while the program is running, corresponding alarms/error messages are transferred to data block DB2. 6 **alarms** are implemented:

1. Error in HMI bus address, HMI bus type
2. No confirmation HMI 1 offline
3. HMI 1 is not going offline
4. No confirmation HMI 2 offline
5. HMI 2 is not going offline
6. Requesting HMI is not going online

and an **error message**: Sign-of-life monitoring error

If the default settings in FB101 are not changed, the alarms start at

DB2.DBX188.0 (1st alarm) and end at

DB2.DBX188.(6th alarm)

With variable:

DBX_Byte_alarm, the byte value for the 6 alarms can be changed from the default setting of 188. With variable:

DBX_Byte_report, the byte value of the operational message can be changed from the default setting of 192.

Mixed mode

Definition:

The term "mixed mode" refers to a state in which a conventional OP without control unit switchover function is connected to the first HMI interface on the NCU. The control unit switchover then operates exclusively on the 2nd HMI interface.

Parameter "MMC_mixed_mode" (variable in FB101/DB101) can be set to switch the operating mode from pure HMI operation to mixed mode by entering the following settings.

When FB101 is called, this must be assigned:
TRUE Mixed mode is effective. Control unit switchover operates on the 2nd interface
FALSE Mixed mode is not effective. Control unit switchover operates on both interfaces

Supplementary conditions:

- A machine control panel (MCP) must not be configured for the first HMI interface. The first HMI interface is always assigned **active** mode status in mixed operation.
- To allow the second HMI interface to be assigned active mode status, it is possible to assign active mode status to both online interfaces in mixed operation. However, certain supplementary conditions apply.

⚠ WARNING

When data are input from both control units simultaneously, there is a risk that inconsistent data will be transferred to the control.

Server mode

Once a server has occupied the HMI interface of an NCU, it cannot be suppressed by any other device.

Processing operating authorization for servers

Three server-related program branches for handling HMI requests are implemented in the control unit switchover program:

1. Request for relinquishing operating focus
2. Request for setting operating focus
3. Relinquish operating focus

Each branch checks/processes the first and second HMI interfaces.

The requests are positively acknowledged if no switchover disabling commands are active. The "Relinquish operating focus" includes deactivation of the relevant machine control panel.

See also diagrams in Section "Examples/Graphic representation of function sequences".

Wait times for acknowledgement signals

To render the program independent of timers, two wait times based on repeated reading of the system time are implemented via SFC64 in the control unit switchover program. The wait times for acknowledgement signals can be changed if necessary by means of:

FB101/DB101 variables:

waiting_period_1: Wait for activation/Online HMI

waiting_period_2: Wait for deactivation/offline HMI

Wait for HMI sign of life:

Values can be assigned to the FB101 variables in the range:

0-32 (seconds). These values are entered in ms.

Program integration

If the control unit switchover program on the PLC is to be called as a function in a higher-level PLC program, then it must be ensured that FB101, FB102 and FC103 and associated instance data blocks DB101, DB102 have not already been used elsewhere.

Initialization

When the NCU is restarted, all signals relating to control unit switchover on the PLC interface in DB19 are set to zero.

Note

Before an NCU is initialized, it must be ensured that it is **not** currently linked **online** to any PCU/HT6.

It may be necessary to perform an HMI restart.

Resetting of interface by PLC

The interface signals relating to control unit switchover can be reset selectively as follows (without RESET on the NCU):

FB101/DB101 variables:

Initialization

TRUE: Single reset of signals in DB19. After signal reset, the Initialization parameter is automatically reset to FALSE.

Sign-of-life monitoring

Once a PCU/HT6 is connected online, it sends a sign of life in DB10 DBB108. If a PCU/HT6 in online mode does not send a sign of life signal for longer than the time set in waiting_period_2, then the PLC program generates the message: "Sign-of-life monitoring error". This message is not canceled until one of the PCUs/HT6s is switched from offline to online mode again.

Identifier for PCU/HT6 "Control unit switchover exists"

In certain operating states, PCUs/HT6s must be able to detect whether the control unit switchover function exists. The requesting interface signal in DB19.DBW110 **m_to_n_alive** is provided for this purpose. As soon as block FB101 is called in the PLC, it also sends a sign of life signal, consisting of the cyclic incrementation of **m_to_n_alive** (ring counter).

Generation after adaptations

When static parameters are changed in FB101,

- DB101 must be deleted,
- regenerated,
- called and
- stored.

Blocks and functions used

Function blocks	**FB101, FB102**
Instance data blocks	DB101, DB102
Functions	FC103
DB of interface	DB19
Global data block for error messages	DB2, (DB3)
Timer auxiliary function	SFC64

2.3 Several operator panel fronts and NCUs, standard functionality

The M:N concept **without** the **Control Unit Management** option is described below.

Note

This section does not apply to the HT6, since only one HT6 can be operated on an NCU without control unit management.

2.3.1 Configurations

Configuration parameters

As it is possible to freely combine hardware components, it is necessary to inform the system which components are combined and in what manner. On the HMI Advanced, this is done by means of an operator dialog in the commissioning area. In the case of the HMI Embedded/OP030, the configuration parameters are entered through the creation of a configuration file which is loaded for commissioning. The file must be structured as described below.

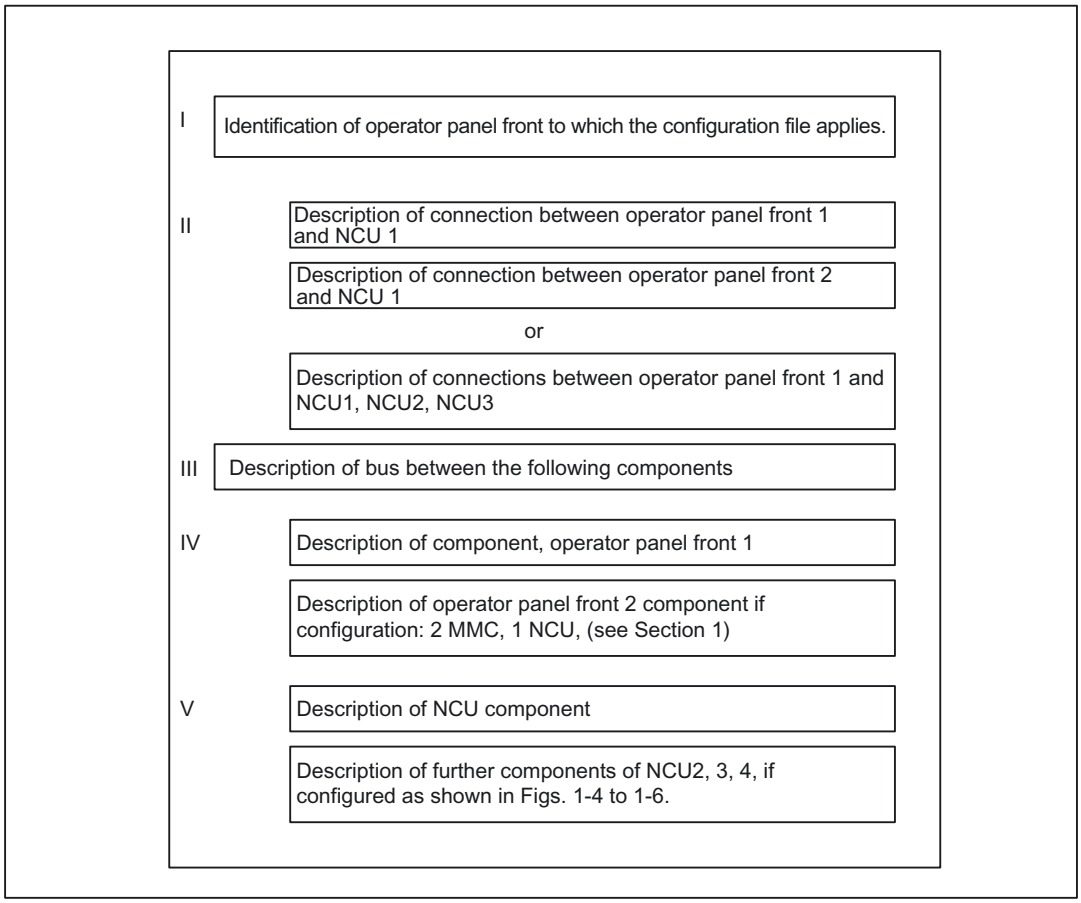


Figure 2-14 Structure of configuration file NETNAMES.INI

Examples

For complete examples of configuration files, please refer to Section "Examples" in this description.

Syntactic declarations

The configuration file must be generated as an ASCII file. The syntax is the same as that used in Windows *.ini" files.

In the following tables, the components which the user may need to adapt are printed in *italics*. Alternative parameters are specified separated by a |. Parameters must be entered in lower-case letters. Comments can be entered, provided they start with ";" and are limited on the right by the end of line. Blanks may be used as separators at any position except for in identifiers and parameters.

Number of configuration files

A configuration file is required for every operator panel connected.

The configuration files of different operator panels in a configuration differ from one another only in the first entry, which contains the assignment of the file to a specific operator panel ([own] see below). For practical purposes, the core of the file is generated only once and then copied for the second operator panel. The identifier of the operator panel to which the file applies is then inserted in the copy.

I. Identification of operator panel

Identification of operator panel to which the configuration file applies.

Table 2- 1 Identification of operator panel front

Descriptive entry	Formal	Example
Header	[own]	[own]
Next line	owner = *Identifier*	owner = MMC_2

Identifier: A descriptive entry for an operator panel must be generated with the selected identifier according to IV.

Vocabulary words:
own:Introduces identification entry
owner

II. Connections

Description of connections from the operator panel components to the NCU to be addressed. An entry of the following type is required for each operator panel.

Table 2- 2 Description of connections

Descriptive entry	Formal	Example
Header	[conn *Identifier*]	[conn MMC_1]
Next line(s)	conn_i = *NCU_ID*	conn_1 = NCU_1

Identifier: A descriptive entry for an operator panel front must be generated with the selected identifier according to IV.

*NCU_ID:*A descriptive entry for the NCU must be generated with the selected NCU identifier according to V.

Vocabulary words:
conn: Introduces connection entry
conn_i: Password for connection, i = 1, 2, ..., 8.

III. Description of bus

The hardware allows links to be implemented via different buses which are differentiated mainly by their baud rates. The bus type used must be specified.

Table 2- 3 Description of bus

Descriptive entry	Formal	Example
Header	[param network]	[param network]
Next line	bus = opi | mpi	bus= opi

Vocabulary words:
param network Introduces descriptive entry for network
bus: Bus
opi: Operator panel front interface with 1.5 Mbaud
mpi: Multi Point Interface with 187.5 Kbaud

Note

The baud rate is automatically detected on the HMI Embedded.

IV. Description of operator component(s)

A separate entry must be generated for each operator panel connected to the bus. A maximum of two entries in SW 3.x.

Table 2- 4 Description of operator component

Descriptive entry	Formal	Example
Header	[param *Identifier*]	[param MMC_1]
Next lines (optional)	name= *any_name*	name = MMC_A
(optional)	type= *mmc_100* | *mmc_102* | *op_030*	type = mmc_100
	mmc_address = j	mmc_address = 1

Identifier: Entry for first or second operator panel

bel_name: User-defined name with a maximum of 32 characters

mmc_100 | *mmc_102* | *op_030*: Operator component type

j: Address of operator component on the bus: *j* = 1, 2, ... 31

Vocabulary words:
param: Introduces parameter for operator component
name: User-defined name for the operator component to be described
type: Operator component type
mmc_address: Bus address of operator component

V. Description of NCU component(s)

A separate entry must be generated for each NCU component connected to the bus.

Table 2- 5 Description of NCU component

Descriptive entry	Formal	Example
Header	[param *NCU_ID*]	[param NCU_1]
Next lines (optional)	name= *any_name*	name= NCU1
(optional)	type= *ncu_561* | *ncu_571* | *ncu_572* | *ncu_573*	type= ncu_572
*)	nck_address = *j*	nck_address = 13
*)	plc_address = *p*	plc_address = 13

NCU_ID: Entry for NCU component.

bel_name: User-defined name with a maximum of 32 characters; with HMI Advanced, the name entered here (e.g. NCU1) is also output in the alarm line.

ncu_561|ncu_571|ncu_572|ncu_573: NCU type, (ncu_561 not with configuration 1 PCU, 3 NCU)

j: Address of NCU component on the bus: *j* = 1, 2, ... 31 *)

p: Address of PLC component on the bus: *p* = 1, 2, ... 31 *)

If bus = opi, the same value must be entered for *j* and *p*.

***)** If bus = mpi, the following applies: Since the associated NCU always occupies the next-higher address than the PLC, the PLC address must not be 31. Address 31 can, for example, be assigned to a PCU.

Vocabulary words:
param: Introduces parameter for NCU component
name: User-defined name for the NCU to be described
type: NCU type
nck_address: Bus address of NCU
plc_address: Bus address of PLC

Note

If the bus node addresses on the MPI bus are configured in conformance with SIMATIC, the configuring engineer can read out the assigned addresses using a SIMATIC programming device and use them to create the NETNAMES.INI file.

Defaults

The following defaults are applied if no NETNAMES.INI configuration file has been copied into the HMI Embedded/OP030 or if the file cannot be interpreted:

- The bus type used is detected automatically.
- PCU has address 1.
- OP030 has address 10.
- NCU and PLC both have address 13 for an OPI bus.
- NCU has address 3 and PLC address 2 for an MPI bus.

If the network configuration actually corresponds to these default settings, then it is not necessary to explicitly generate and load the NETNAMES.INI file. If a special file is created, however, it must correspond exactly to the network configuration.

2.3.2 Switchover of connection to another NCU

Note

The channel menu function is an option and must be configured in the NETNAMES.INI file.

You can change to the channel menu in all operating areas by activating the channel switchover key. The only change is to the horizontal and vertical softkeys.

Use the horizontal softkeys to select a channel group (max. 24), 8 links to channels on different NCUs can be set up in each channel group.

The "Channel menu" display shows all the current communication connections and the associated symbol names.

2.3.3 Creating and using the configuration file

HMI Embedded, OP030

The NETNAMES.INI ASCII file generated on the PC or programming device is loaded via the RS-232 interface and permanently stored in the FLASH memory of the control units.

References: FBO/IK/, Installation Kit

HMI Advanced

The NETNAMES.INI file can be edited directly with an editor (in menu "Commissioning/HMI/Editor" or DOS_SHELL) on the hard disk of the operator component. The NETNAMES.INI file is stored in the installation directory:

C:\USER\NETNAMES.INI.

2.3.4 Power up

Differences between HMI Embedded and HMI Advanced

Due to the differences in power-up characteristics, different commissioning procedures are required.

- HMI Embedded always runs in "M:N" mode, when "M:N" is configured in the NETNAMES.INI file.
- The mode can be set in the "Commissioning/HMI/Operator panel" menu on the HMI Advanced. The HMI Advanced always runs in a "1 : 1" link with an NCU, the NCU address can be specified directly. If "M:N" mode is set, then HMI Advanced searches the NETNAMES.INI file for the names of the specified partners. The addresses are freely assignable.

  Recommendation:

  Reserve address 0 (for PG)

  Reserve address 13 (for service case: NCU replacement)
- The OP030 does not have the functional capability "M:N". It can be used as a second operator panel front that is permanently assigned to an NCU("1 : 1" link). The addresses of the connected partners can be set for this purpose.

Note

It is advisable to make a written record of the procedure (address assignments, etc.) beforehand.

Commissioning

The NCUs are assigned bus address 13 in the delivery state. Every NCU on the bus must be allocated its own, unique bus address.

Addresses are assigned in:

- HMI: NETNAMES.INI file
- NCK: In the menu "Commissioning/NC/NCK address"
- MCP: Switch... (address and possibly baud rate, see also /IBN/)
  OB100 parameters: ...(see also FB1/P3).

Note

An NCK address is not deleted with "Delete SRAM" (switch S3= position "1" on NCU).

Power up with HMI Embedded/HMI Advanced

See Section "Several operator panel fronts and NCUs with control unit management option/power up".

2.3.5 NCU replacement

The procedure for NCU replacement or configuration of an additional NCU is similar to that for commissioning (see "Power up").

Variant 1

1. Establish 1:1 connection between PCU and NCU
2. Power up HMI on NCU with bus address "13" (see above)
3. Enter new NC address via the Commissioning/NC/NC address operating area and boot NCU.
4. Rewire bus for M:N operation

Variant 2

1. The NCU which is the "power-up NCU" for a PCU connected to the bus is disabled. (The HMI powers up at the first connection configured in the NETNAMES.INI file)
2. Power up HMI on NCU with bus address 13 (see above)
3. Enter new NC address via the Commissioning/NC/NC address operating area and boot NCU.
4. Reactivate "Power-up NCU" again

Note

Please note:

- Bus address 13 must be reserved for servicing purposes (and should not be assigned to a bus node).
- HMI Embedded:

  The length of the names in the file NETNAMES.INI (configuration of channel menu) is limited to five characters.
- HMI Advanced:

  The "mstt_address" data item is not evaluated, it is used for documenting the bus nodes. If the channels are located on different NCUs, "m:n" must be entered in the Commissioning/HMI/Operator panel operating area.

Data exchange between NC<->PLC

In configurations consisting of 1 x PCU and n x NCUs, it is often necessary to synchronize the NCUs.

The following synchronization options are available:

- NCK I/Os on drive bus (digital, analog, writing of NC and PLC).
- PLC I/Os (I/O link).
- Link via PROFIBUS DP.
- Link via Global Data function of SIMATIC S7.

2.4 Restrictions for switchover of operator components

Rejection of link

On switchover to another NCU, the NCU selected for the new link may reject the connection. There may be a defect in the NCU or no further operator panel can be accepted. In this case, the HMI Embedded automatically switches over to connection 1 after approx. five seconds. HMI Advanced displays "#" for the variables.

Alarms, messages

The operating characteristics depend on the HMI type:

1. HMI Embedded/OP 030

   Due to the equipment restrictions on driver level and the limited working memory, only the alarms/messages of one NCU can be processed at any one time.

2. HMI Advanced

   Only one alarm text file is managed. The NCU name assigned in the NETNAMES.INI file is displayed as the NCU identifier in front of every alarm or message. To obtain user texts specific to the NCU, it is possible to define user areas in the PLC for certain NCUs. The alarms/messages of all connected components can be processed and displayed simultaneously.

Operator interface

The operating characteristics depend on the HMI type:

1. HMI Embedded

   Only fields and variables of one NCU can be displayed simultaneously in a window. Only the alarms and messages from the NCU which is currently connected to the PCU are displayed.

   Up to four connections (one active connection, three other connections) can be displayed simultaneously via user configuration (OEM). All the variables (alarms and messages) of a connection must be contained in one window (window-specific connections).

2. HMI Advanced

   Fields and variables of different NCUs can be displayed in the same window (as an OEM application). Alarms and messages from all the NCUs connected to the PCU can be displayed.

3. OP030

   OP030 can only be configured as a "1 : 1" connection to an NCU.

When the HMI Embedded and HMI Advanced are used in the standard configuration (Section "Configurability"), it is not necessary to configure the operator interface. If variables of different NCUs need to be output simultaneously in a display, configuration is necessary.

References:
/BEM/ SINUMERIK HMI Embedded/UOP Configuration kit

2.5 NCU link

2.5.1 Introduction

The number of channels or axes per NCU is restricted due to the limitation on memory and computing capacity elements. A single NCU is not sufficient to fulfill the requirements made by complex and distributed machines, such as multi-spindle and rotary indexing machines. For this reason, the control system and closed-loop axis controls are distributed among several NCUs.

In order to ensure, however, that channels and axes can continue to operate on an interrelated, cross-NCU basis, the system provides NCU link functionality.

Function extensions

The following applications are possible:

- Cross-NCU interpolation (coupling of setpoints, actual values and VDI signals)
- Real exchange of axes
- Cross-NCU access to axis values and axial system variables
- NCU-user communication supported by NCU link variables
- Generation of alarms on the NCU affected by an irregularity, even if the cause of the problem is on another NCU.

2.5.2 Technological description

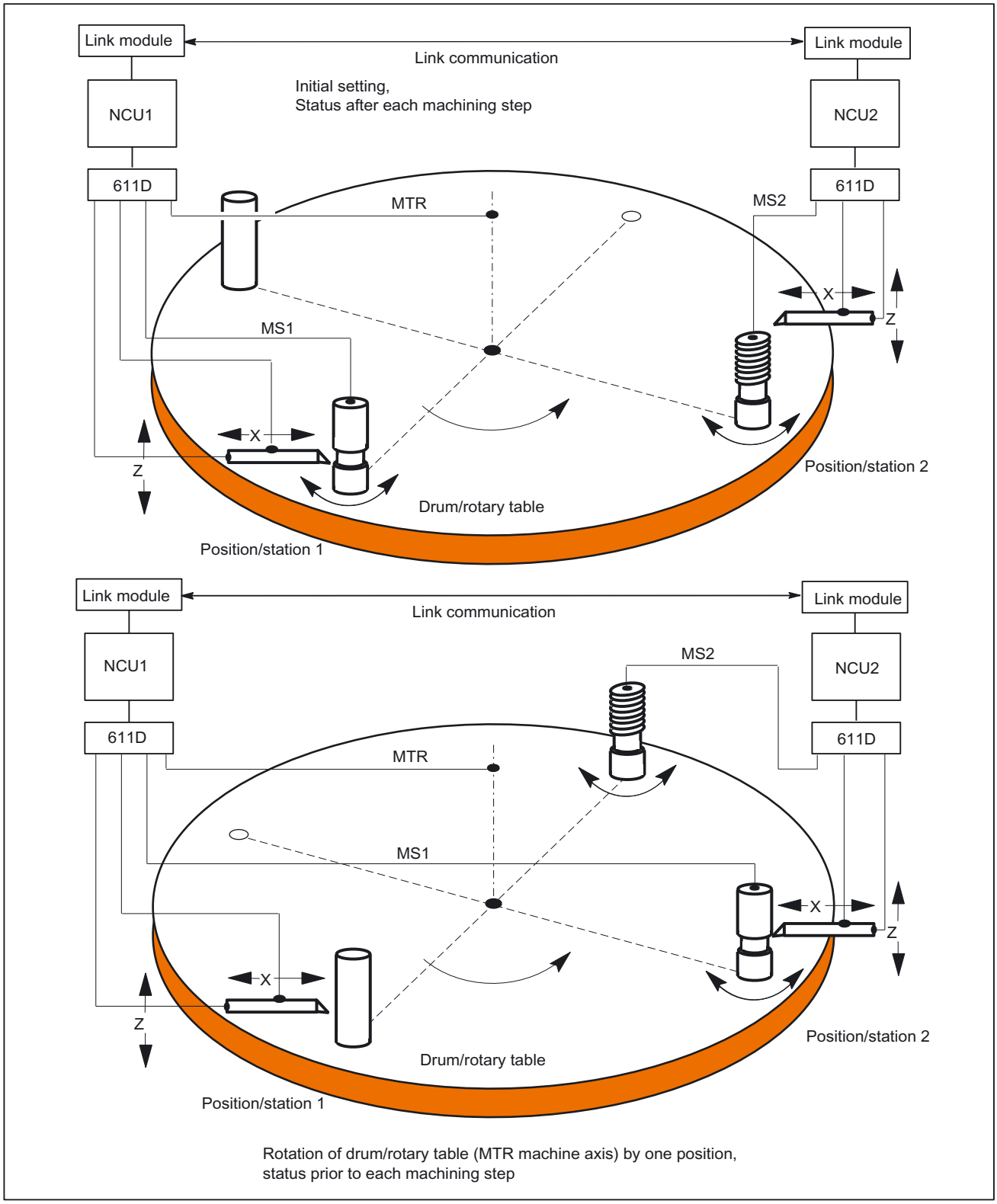


Figure 2-15 Sectional diagram of a drum changeover

The diagram shows the main components of a simple multi-spindle plant. Several spindles are mounted mechanically on the drum, each of which can used to perform a different machining operation. Together with the slide (X and Z axes), they form a machining station which is assigned to one channel. A workpiece is rotated by one spindle.

The workpiece to be machined is loaded and unloaded only once. The tool is mounted on the slide (e.g. X, Z axes). Various different tools can be loaded for each machining operation.

The tool is continuously assigned to the machining station. The workpiece-holding spindles are moved from one machining station to the next.

The spindle can only be checked for the current machining process. The channel must be able to address the slide axes and the current spindle at any given time. Every time a spindle moves on to the next machining unit, however, the spindle addressed by the channel must be a different machine axis. The "**axis container**" concept solves the variable imaging of channel axes on machine axes. The machine axes might belong to another NCU connected by means of the NCU link. An accessible machine axis belonging to another NCU is referred to as a **link axis** (see "Link axes").

The following subjects are closely related to the NCU link function and are dealt with in separate subsections.

- Link axes
- Axis container
- Cross-NCU user communication
- Configuration of the link grouping

2.6 Link axes

Introduction

This subsection describes how an axis (for example, B1 in diagram "Overview of link axes"), which is physically connected to the drive control system of NCU2, can be addressed not only by NCU2, but also by NCU1.

Requirements

- The participating NCUs, NCU1 and NCU2, must be connected by means of the link module.

References: /PHD/ 840D NCU Configuration Manual, Link Module

- The axis must be configured appropriately by machine data.
- The link axis option must be installed.
- Link communication must be activated with machine data:

  MD18780 $MN_MM_NCU_LINK_MASK

  The link grouping must be configured as described in "Configuration of link axes".

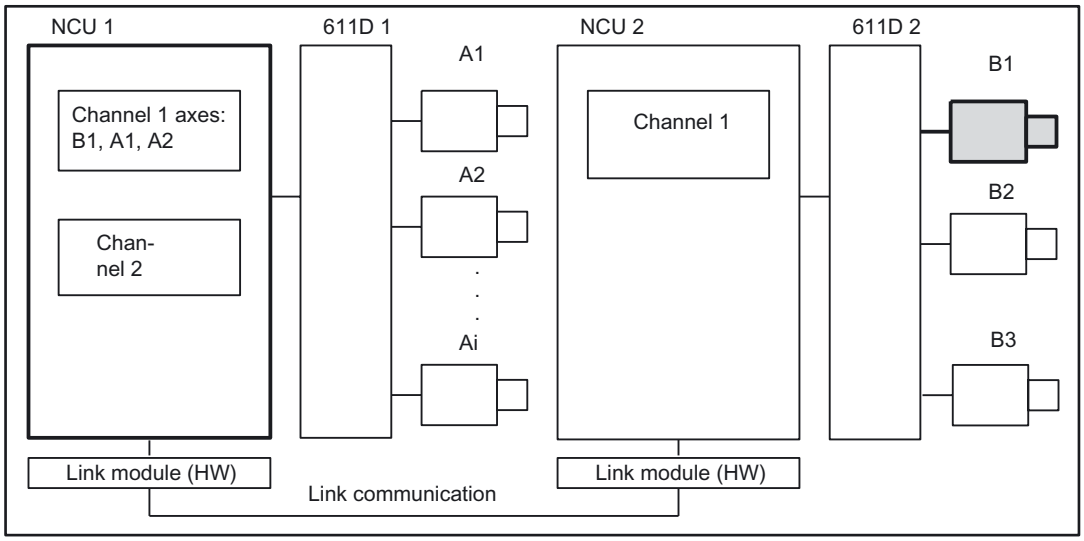


Figure 2-16 Overview of link axes

Terminology

The following terms are important for understanding the subsequent description:

- **Link axis**

  Link axes are machine axes, which are physically connected to another NCU and whose position is controlled from this NCU. Link axes can be assigned dynamically to channels of **another** NCU. From the standpoint of a particular NCU, they are not → local axes. Dynamic changes in the assignment to a channel (exception: channel on another NCU) are implemented based on the concept of **axis container** described under "Axis container".

  Axis exchange with `GET` and `RELEASE` from the part program is only available for link axes within an NCU. In order to cross the NCU limit, the axis must first be placed in the NCU or a channel using the axis container function so that it can then be exchanged optionally in the same way as any other axis.

- **Local axis**

  A local axis is only addressed by the NCU to whose drive bus it is connected.

- **Link communication**

  The link communication is implemented by link modules on the NCUs involved. The link communication consists of setpoints, actual values, alarm handling, global variables (data) and signals (axis signals, PLC signals).

- **Home NCU**

  The NCU which sets up the drive bus connection for a → link axis and which performs position control is referred to as the home NCU.

  In the diagram "Overview of link axes", NCU2 is the home NCU for → link axis B1.

- **Interpolation**

  The **link axis** option enables interpolation between → local axes and axes on other NCUs for NCUs with → link communication..

  If the interpolation is not only local, cyclical data exchange (setpoints, actual values, ...) takes place within an interpolation cycle. This results in dead time, in particular when waiting for external events.

- **Axis change**

  Use of a → link axis by a specific NCU can change dynamically. The **axis container** mechanism described under "Axis container" is provided for this purpose. The part program command `GET` is not available for link axes, and the part program command `GETD` is only available on the same NCU.

  With software version 4 it was only possible to exchange axes between different channels of the same NCU.

- **Configuration of link axes**

  NCUs that want to use → link axes must configure the **NCU identifiers** for the home NCU of the link axis in addition to the usual channel and axis machine data.

- **Home channel**

  Channel in which the setpoint-generating part program for the axis is executed after the installation has powered up.

- **Lead link axis**

  From the point of view of an NCU (2) that traverses following axes, a *leading axis* that is traversed by another NCU (1). The required data for the master value axis are supplied via → link communication for NCU (2). Axis coupling between the leading axis and the following axis/axes is implemented, for example, by means of a curve table.

2.6.1 Configuration of link axes and container axes

Function

The link function makes it possible to increase the number of axes and channels in the installation. The necessary link cycle increases with the number of link axes and container axes.

Typical link cycles are:

- 2 NCUs with 2 containers, each with 6 axes or spindles: 8ms
- 3 NCUs with 2 containers, each with 6 axes or spindles: 17ms
- 4 NCUs with 2 containers, each with 4 axes or spindles: 17ms

Note

The times increase if more link variables have to be transferred in addition.

Machine axis image

The channels operate with one of 31 logical axes from the logical machine axis image. This image points to:

- Local axes
- Link axes
- Container slots

Container slots in turn point to:

- Local axes or
- Link axes

The diagram below illustrates the interrelationships:

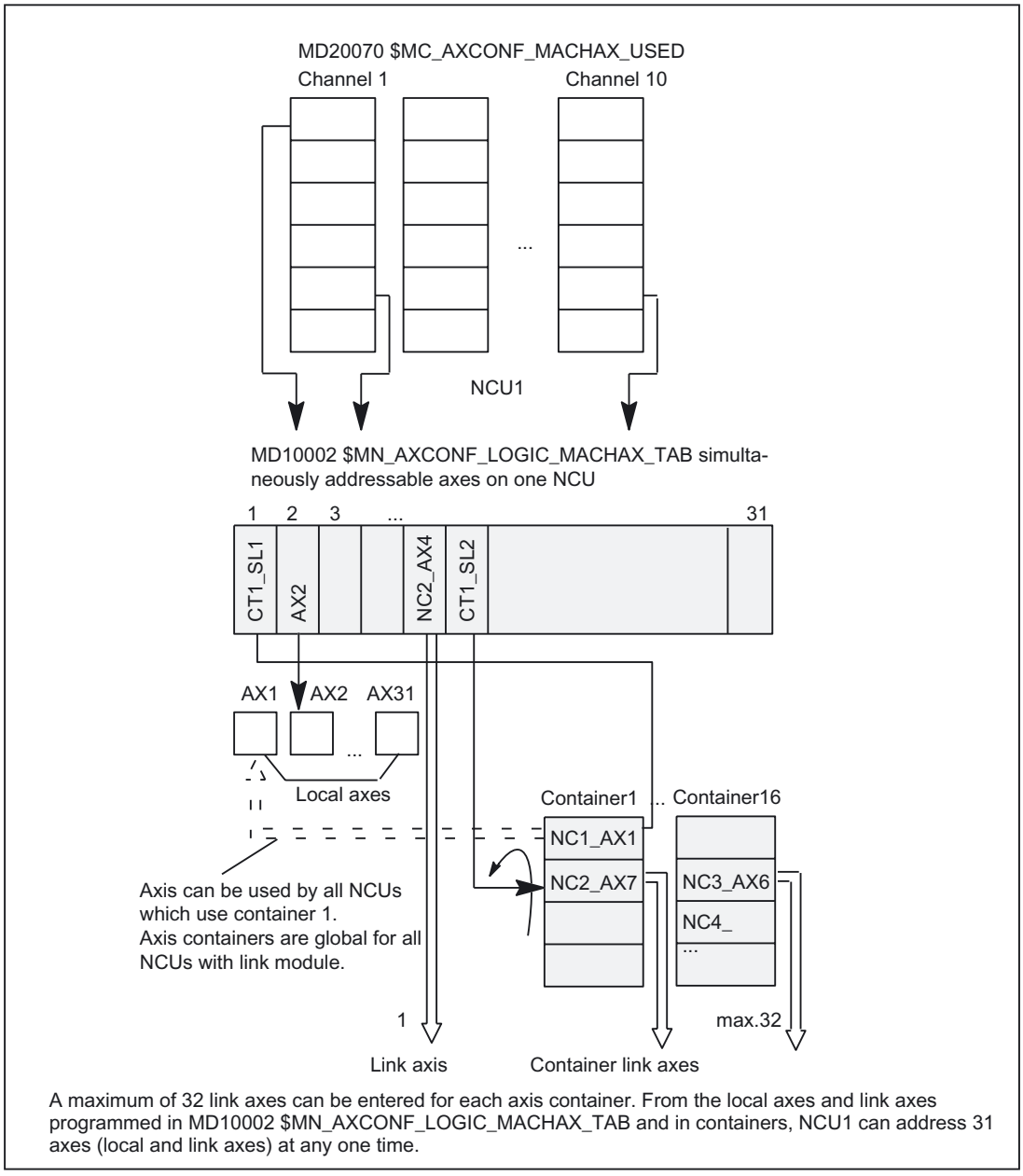


Figure 2-17 Configuration of link axes

Differentiation between local/link axes

To enable link axes to be addressed throughout the system, the configuration must contain information about the axis NCUs. There are two types of NCU axis, i.e. local axes and link axes. The table created by the machine data is used to differentiate between them:

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB.

Note

The **axis container** functions are described in the subsection "Axis container".

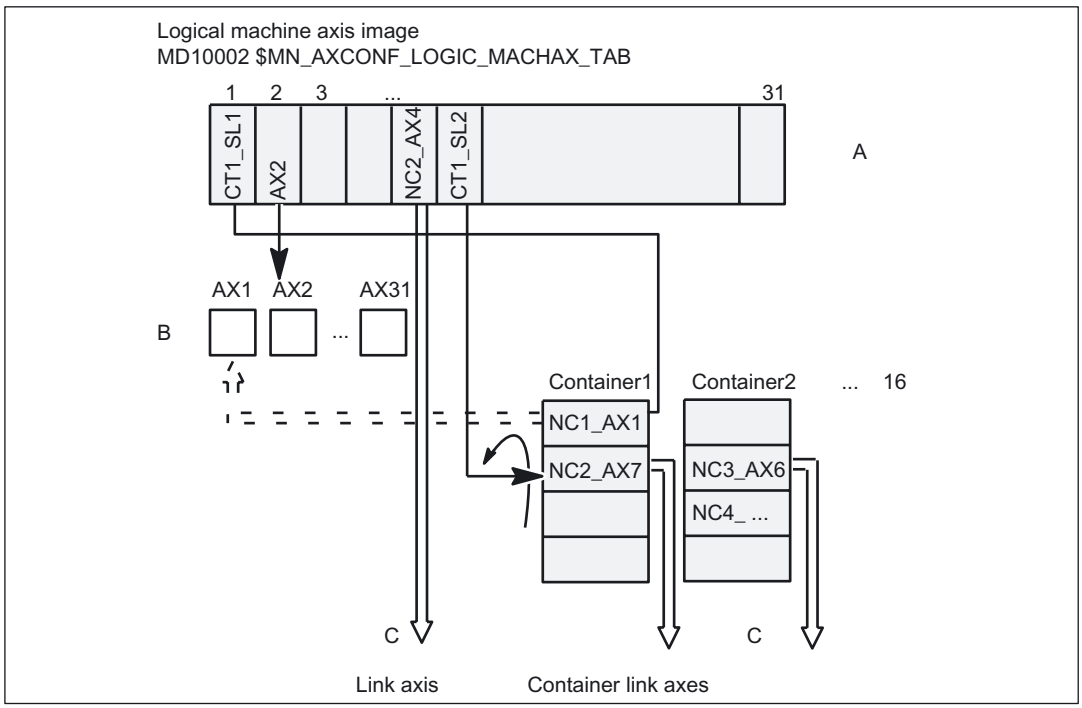


Figure 2-18 Assignment of channel axes to local machine axes and link axes

Explanation

Logical machine axis image A addresses local machine axes B and link axes C.

The number of local machine axes in B is limited. The maximum permissible number for a specific system can be found in Catalog NC60.1.

All axes that can address the NCU are contained in B and C together.

Entries in A have the following format:

$MN_AXCONF_LOGIC_MACHAX_TAB[n] = NCj_AXimit

n: Index in Table A

NC: stands for **NCU** with

j: **NCU number**, 1 <= j <= 16

i: Axis number, 1 <= i <= 31

Channel axes are assigned to **logical machine axis image A** via machine data:

MD20070 $MC_AXCONF_MACHAX_USED

Viewed from the part program, the only accessible machine axes are those which can be addressed by the channel (possibly via axis container, see below) via the logical machine axis image at a given point in time.

Default

By default, the settings of logical machine axis image A are local axis name AX1 for entry 1, and local axis name AX2 for entry 2, etc.

Examples

The following expressions can appear in the logical machine axis image, for example:

NC2_AX7: Machine axis 7 of NCU 2

AX2: Local machine axis 2

If expressions exclusively of the form AXi above are entered in the logical machine axis image, a configuration is produced which only allows local axes to be addressed.

Notice: The defaults are as follows:

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[0] = AX1

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[1] = AX2

...

Note

Another valid format for entries in the logical machine axis image A is:

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[n] = CTx_SLy where

CT: stands for **container**

x: Container number, 1 <= x <= 16

SL: stands for **slot**

y : Slot number, 1 <= y <= 32

Axis containers are an arrangement of axes in groups which can be changed dynamically. Axis containers are described in Section "Axis container".

2.6.2 Axis data and signals

Introduction

Axis data and signals for a link axis are produced on the home NCU of the link axis. The NCU that has caused the movement of a link axis is provided with axis data and signals from the system:

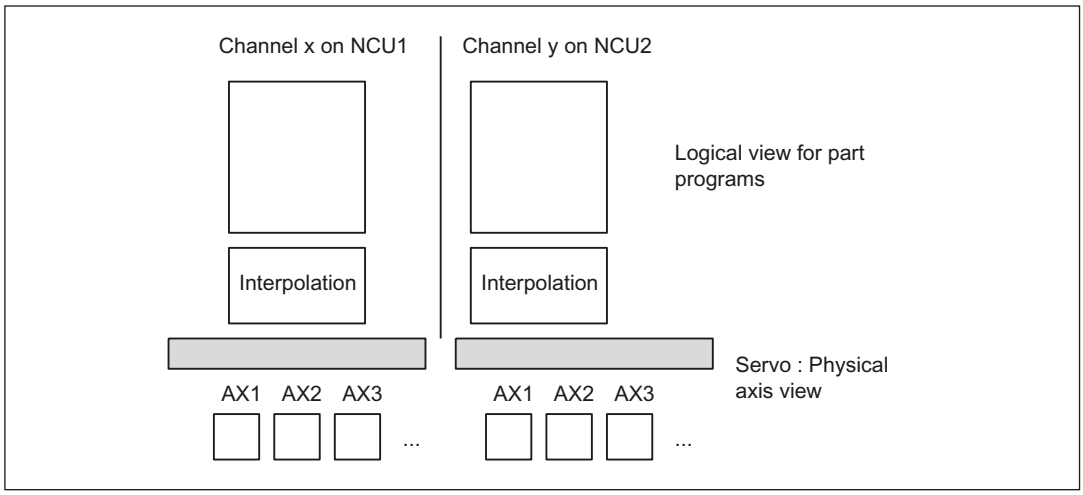


Figure 2-19 Views of axes

Implicitly active link communication

During interpolation, data are made available for axes which are physically subordinate to a non-local servo (identifiable from entries in machine data:

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB or axis container),

via the link communication in the same manner as they are provided for local axes from the logical viewpoint of part programs. The procedure remains concealed from the applications.

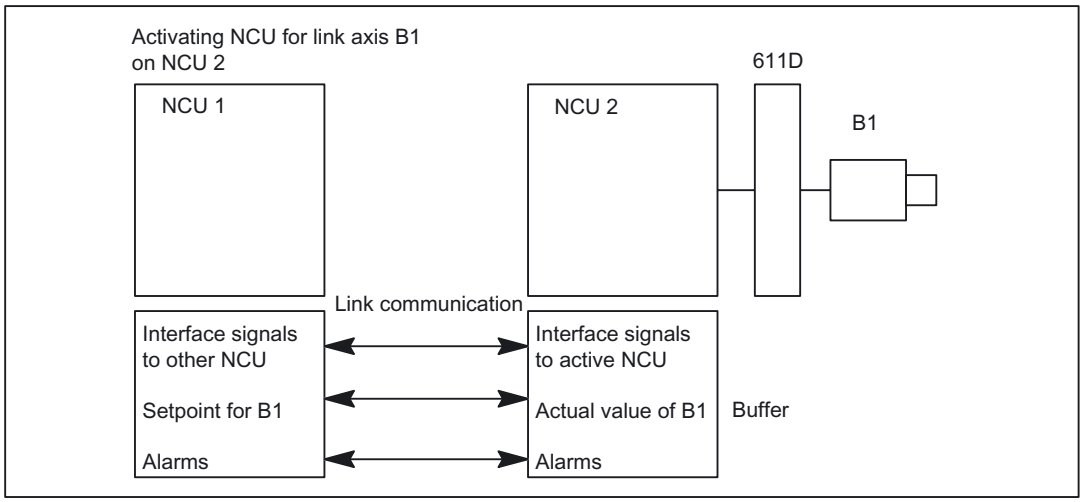


Figure 2-20 Exchange of operating data and signals of a link axis

Position control

The position control is implemented on the NCU on which the axis is physically connected to the drive. This NCU also contains the associated axis interface. The position setpoints for link axes are generated on the active NCU and transferred via the NCU link.

Communication methods

There are two types of link communication:

- Cyclic communication
- Non-cyclic communication

Cyclic communication

- Setpoints for link axis
- Actual values of link axis
- Status signals of link axis
- Status signals of NCUs

are transferred cyclically. Actual values and status signals of a link axis are updated and made available to the NCU that is interpolating this axis.

Non-cyclic communication

- Exchange of link variables
- Warm restart requirements
- Activation of axis container rotation
- Changes in NCU global machine and setting data
- Activation of axial machine data for link axes
- Alarms

Transfer time

Delays are incurred for transferring **setpoints** to the home NCU of a link axis and for returning the **actual values** of this axis. With an interpolation group of local axes and link axes, the control delays the setpoints for the local axes of the interpolating NCU by one interpolation cycle, such that consistent values are taken into account for the interpolation.

If a channel needs the actual values of an axis of another NCU, e.g. a spindle with thread cutting, two interpolation cycles will elapse before they are available. The setpoints then generated are sent one interpolation cycle later to the position control for the above reason.

Response of the AXIS-VAR server to errors

If the server cannot supply any values for an axis (e.g. because the axis concerned is a link axis), then it returns a default value (generally 0).

For the purposes of testing, the machine data of the axis data servers below can be set to sensitive, with the result that it returns an error message instead of default values:

MD11398 AXIS_VAR_SERVER_SENSITIVE

0: Default value

1: Error message

2.6.3 Output of predefined auxiliary functions in the case of an NCU link

Predefined M, S, F auxiliary functions

For link axes and container link axes, a predefined M, S, and F auxiliary function is transferred from the NCK via the NCU link to the home NCU of the link axes and output from there as system auxiliary functions to the PLC. These system auxiliary functions are evaluated by the PLC and output as follows:

DB31, ... DBW86 (M function (INT) for spindle)

DB31, ... DBD88 (S function (REAL) for spindle)

DB31, ... DBD78 (F function (REAL) for axis)

Note

The transfer from the NCU link to the home NCU is only relevant for predefined spindle auxiliary functions M3, M4, M5, M19 and M70.

For more information about "Predefined auxiliary functions", see

References: /FB1/ Function Manual, Basic Functions; Auxiliary Functions to PLC (H2)

Alarm 14768

If the system auxiliary functions received via the NCU link cannot be output via the VDI interface, because, for example, the transfer buffer is full, alarm 14768 "Cannot output axial auxiliary functions received via NCU link" is issued.

Examples

An NC program with M3 S1000 is executed for the 7th channel on NCU_2. This spindle corresponds to the 5th machine axis of NCU_1 and is therefore link axis. Therefore the auxiliary function output here for NCU_2 is in Channel 7 with the axis number 0, as the link axis is on another NCU (NCU_1 here). On the PLC of NCU_2, this results in the output of:

DB21, ... DBW68 (extended address of the M function (16-bit INT))

DB21, ... DBD70 (M function 1 (INT 3))

DB21, ... DBW98 (extended address of S function 1)

DB21, ... DBD100 (S function 1 (REAL 100))

In parallel, the information of the system auxiliary functions is transferred from NCU_2 via the NCU link to NCU_1 (home NCU of the link axis). The system auxiliary functions M3 S1000 for the 5th machine axis are output from here.

The PLC on NCU_1 then supplies the following axial signals on the PLC user interface for machine axis 5:

DB35, ... DBW86 (M function (INT 3) for spindle)

DB35, ... DBD88 (S function (REAL 100) for spindle).

2.6.4 Supplementary conditions for link axes

Output of alarms from position controller or drive

Axis alarms are always output on the NCU which is producing the interpolation value. If an alarm is generated for a link axis by the position controller, then the alarm is transferred to the NCU which is currently processing the interpolation.

On the assumption that axis alarms which cause the NCK-Ready relay to drop out (Nck-NoReady) are attributable to faults on the drive bus, the alarm is also output on the NCU on which the axis or the drive bus is physically operated. The reaction "Ready relay dropout" is only activated on this NCU.

Output of alarms following EMERGENCY STOP

If an EMERGENCY OFF request is activated by the PLC on an NCU, all axes physically connected to drives on this NCU are switched to follow-up mode. This means that: even axes which are being interpolated by a different NCU are also switched to follow-up. Since this status prevents any further constructive machining operations on the other NCUs, an additional alarm is generated which is designed to stop all axis motion instantaneously.

This additional alarm must be acknowledged by an NC reset. If the original alarm is still active at this time, then the additional alarm can be successfully reset, but another alarm (self-clearing) is then produced which prevents axis motion or a new program start until the original alarm has been reset.

Output of alarms with alarm reaction NCK-NoReady

If a serious alarm resulting in dropout of the NCK-Ready relay is activated on an NCU, then the effects of the alarm will apply to all other NCUs which are addressing an axis via link communication on the first NCU. An additional alarm which causes all other axes to stop instantaneously is activated on each of the other NCUs.

For alarm acknowledgement, see EMERGENCY STOP.

Compensation

The following compensations are **not** available for link axes:

- Sag compensation (CEC)
- Leadscrew error compensation (LEC)
- Quadrant error compensation (QEC)

Switching off grouped NCUs

If an NCU in a link grouping is switched off or restarted by an NCK reset, all the other NCUs in the link grouping will also be affected. An alarm is generated on these and the process currently taking place is aborted.

See "Configuration of a link grouping (Page 174)".

Powering up an NCU grouping

If an NCU in a link grouping is restarted by an NCK reset, the other NCUs in the grouping will also execute an NCK reset.

Nibbling and punching

To execute nibbling and punching operations, high-speed inputs and outputs must be connected and parameterized on the "interpolation" NCU (on which the part program is being executed). Commands "High-speed nibbling and punching", e.g. `PONS` and `SONS` are not available for link axes.

Travel to fixed endstop

If an axis container axis is being held against a fixed stop, the axis container cannot rotate. Axes can travel to fixed stops on different NCUs and be subsequently clamped without restriction.

Frames

Link axes may be included in the program commands for frames only if they are geometry axes as well. The command only changes the geometry for the channel in which the axis is currently assigned. A frame command for an axis which is not defined as a geometry axis is rejected with alarm 14092.

Rev. feedrate

Setting data SD43300 SA_ASSIGN_FEED_PER_REV_SOURCE refers to the logical machine axis image and then via this to a machine axis (local or link axis).

2.6.5 Programming with channel and machine axis identifiers

Channel axis identifier

Example:

WHENEVER $AA_IW[Z] < 10 DO ...;Current position of Z axis

Machine axis identifier

Example:

WHENEVER $AA_IW[AX3] < 10 DO ...;Scan current position of machine axis AX3

This method of programming is permitted only if machine axis AX3 is known in the channel at the time of scanning.

Note

System variables which can be used in conjunction with channel axis identifiers are specially marked in the Job Planning Programming Manual (Appendix).

2.6.6 Flexible configuration

Introduction

Rotary indexing machines and multi-spindle machines have special requirements as regards the flexible assignment of channel axes to machine axes.

Requirement profile

When advancing the table of the rotary indexing machine or the drum of the multi-spindle machines, the axes/spindles are brought to a new station or position. The NCU which controls the axes of a station as local axes must be able to address the newly changed axes/spindles. The hitherto addressable axes/spindles can now be discarded for this purpose.

Solution

A configuration of the relevant axes in an axis container specified in machine data enables different machine axes to be located in succession behind a channel axis that remains constant. Advancing the rotary table or drum is performed synchronously with the advancing of the axes entered in the axis container.

Axes in an axis container can also be configured as geometry axes.

Note

The axis container has no mode group reference,

i.e. the workpiece-holding, traveling axis can change from one mode group to another at different machining stations.

2.7 Axis container

Axis container

An **axis container** can be imagined as a circular buffer in which the assignment

- Local axes and/or
- link axes

are assigned to channels.

Container axes

Axes in an axis container are also referred to as **container axes**.

Assignments can be "shifted" ("rotation" of the circular buffer) by means of program commands.

The term axis in this case refers to both axes and spindles.

All machine axes in the axis container must be assigned to exactly **one** channel axis at any given point in time.

Note

Rotation of the drum or rotary table is analogous to the rotation of the circular buffer with the assigned axis entries.

Container link axes

Entries in an axis container can refer to machine axes that are installed in a different NCU than the remaining container axes. Such container axes are also denoted as container link axes.

Reference to axis container

In addition to the direct reference to local axes or link axes, the link axis configuration described in Section "Configuration of link axes" also allows direct references to axis containers in the logical machine axis image.

This type of reference consists of:

- Container number
- a slot (circular buffer location within the container)

The entry in a circular buffer location contains:

- a local axis **or**
- a link axis

(either axis or spindle).

Axis container names

The axis containers can be named with the machine data:

MD12750 $MN_AXCT_NAME_TAB

The axis container names can be used:

- in the axis container rotation commands `AXCTSWE()` and `AXCTSWED()` to address the container to be rotated.
- when scanning the states of axis containers using system variables:

$AC_AXCTSWA[ ]	(Release of axis container rotation in channel)
$AN_AXCTSWA[ ]	(Axis container rotation active?)
$AN_AXCTAS[ ]	(Axis container current rotation?)

Defining the container content

For the axis containers 1 ... n, the ramped up assignment between an axis container slot and a machine axis within an NCU grouping is defined with the machine data:

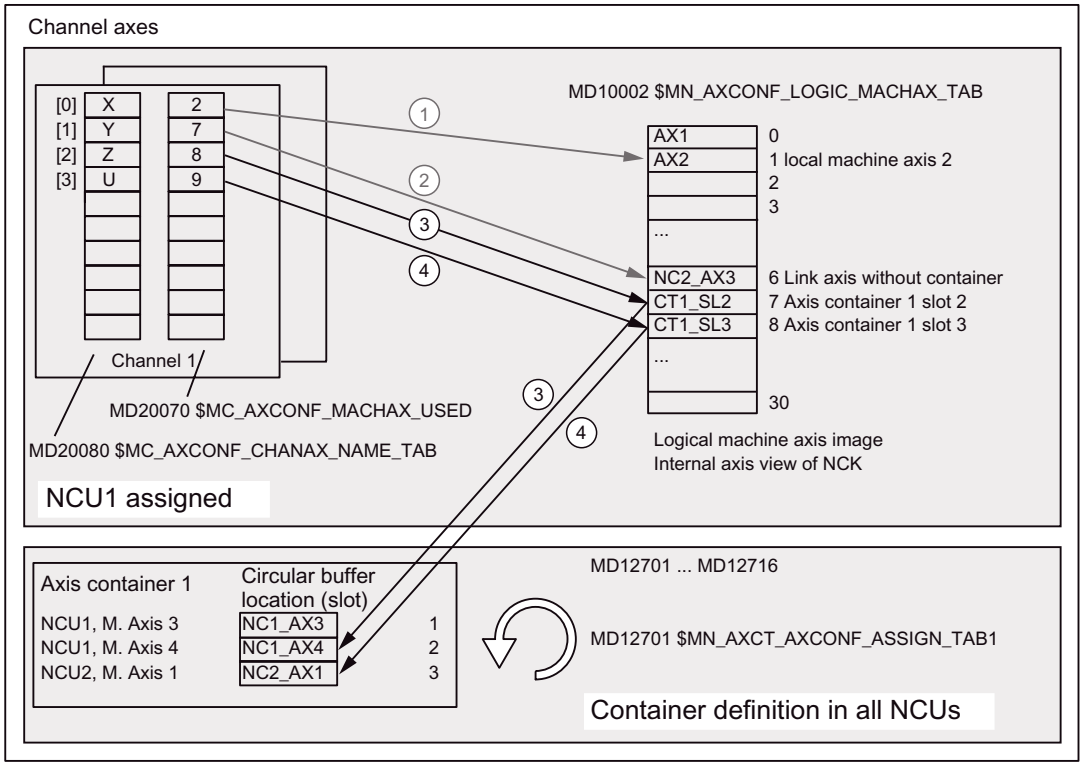
MD12701 ... MD12716 $MN_AXCT_AXCONF_ASSIGN_TAB1...n

The assignment between an axis container slot and the selected channel is programmed in the machine data:

MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB (Logical NCK machine axis image)

Example:



The following assignment is thrown up for the container axes after the control is ramped up (initial state before a first container rotation):

③ **3rd channel axis Z of Channel 1 = 4th machine axis of NCU1**

Explanation:

The 3rd channel axis (MD20070 $MC_AXCONF_MACHAX_USED[2]) shows on the 8th machine axis in the logical NCK machine axis image (MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[7]).

The container slot configuration CT1_SL2 is stored there.

The "NC1_AX4" entry in MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1[1] assigns the 4th machine axis of NCU1 to the 2nd slot of the 1st axis container.

④ **4th channel axis U of Channel 1 = 1st machine axis of NCU2**

Explanation:

The 4th channel axis (MD20070 $MC_AXCONF_MACHAX_USED[3]) shows on the 9th machine axis in the logical NCK machine axis image (MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB[8]).

The container slot configuration CT1_SL3 is stored there.

The "NC2_AX1" entry in MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1[2] assigns the 1st machine axis of NCU2 to the 3rd slot of the 1st axis container. (⇒ **Container link axis**).

The following is applicable to the remaining channel axes shown in the figure:

① **1st channel axis X of Channel 1 = Local machine axis 2**

② **2nd channel axis Y of Channel 1 = 3rd machine axis of NCU2 (⇒ Link axis)**

Container rotation

The contents of the axis container slots are variable inasmuch as the contents of the circular buffer (axis container) can be shifted together by ± n increments.

The number of increments n is defined for each axis container in setting data:

SD41700 $SN_AXCT_SWWIDTH

The number of increments n is evaluated modulo in relation to the number of actually occupied container slots. A new content arises for all the locations of an axis container (exceptions: 0 and slot number = increment number).

System variables provide information about the current status of an axis container; these system variables can be read addressed from the part program and synchronized actions (refer to).

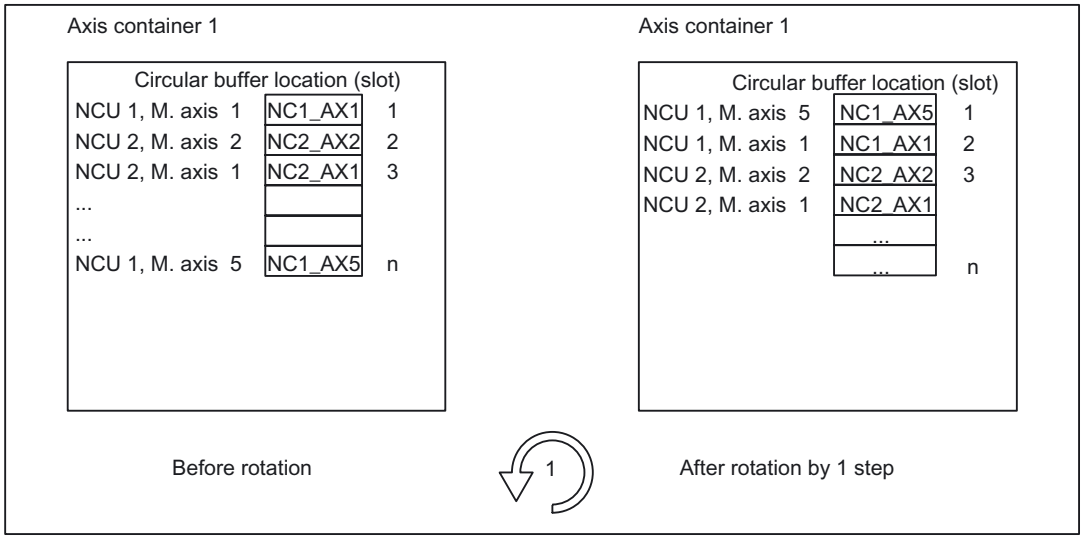


Figure 2-21 Shifting the entries to the axis container slots

The axis container model has the following characteristics:

- A channel always sees a fixed number of axes with defined channel axis names (logical machine axis image)
- The "rotation" of the axis container sets new machine axes for **all** channels that have axes in the same axis container.

Frames in axis container rotations

The assignment between channel axes and machine axes can change when the axis container rotates. The current frames remain unchanged after a rotation. The user himself is responsible for ensuring that the correct frames are selected after a rotation by programming basic frame masks, for example.

Activation of axis container rotation

The application must ensure that the desired local or link axes are addressed by issuing commands in the part program for rotating the axis container to a specific position.

For example, when rotating the drum of a multi-spindle machine into a new position, it must be ensured that each position addresses the newly changed spindle by rotation of the axis container.

Note

Axis containers can be used jointly by different channels of an NCU and by channels of other NCUs.

If axes of different channels display reference to the same axis container via the logical machine axis image, then **all** channels concerned see **different axes** after a rotation. This means: The time for a rotation must be coordinated between the channels. This is performed by means of the available language commands.

Each entry in the axis container must be assigned to the correct channel at all times. The system variables (refer to) offer the possibility for the part program or synchronized action to gain information about the current axis container state.

Commands for the axis container rotation

The requirement outlined above for coordinating channels that jointly use an axis container is contained in the effects of the command AXCTSWE.

Notation:

Programming	Comment
AXCTSWE(CTi)	; The function name represents: AXis ConTainer SWitch Enable CTi is the identifier of the axis container to be moved to the next position/station.

The increment of a circuit must be stored in the following setting data (container-specific):

SD41700 $SN_AXCT_SWWIDTH[container number]

SD41700 is available globally through the NCU global link module (all NCUs connected via link module see the same values).

Function:

Each channel whose axes are entered in the specified container issues an **enable for a container rotation** if it has finished machining the position / station. Once the control receives the enables from **all** channels for the axes in the container, the container is rotated with the increment specified in SD41700 (the direction of rotation is also evaluated based on the leading sign if any):

The variant below is used with a view to facilitating commissioning:

Programming	Comment
AXCTSWED(CT1)	; The function name represents: AXis ConTainer SWitch Enable Direct

The axis container rotates according to the settings in setting data:

SD41700 $SN_AXCT_SWWIDTH[container number]

This call may only be used if the other channels, which have axes in the container are in the **RESET** state.

Note

The axis container names allotted through the following machine data can be used for the `AXCTSWE` and `AXCTSWED` commands:

MD12750 $MN_AXCT_NAME_TAB

Implicit wait

There is an implicit wait for the completion of a requested axis container rotation if one of the following events has occurred:

- Part program language commands which will cause a container axis assigned to this axis container in this channel to move
- `GET`(channel axis name) on a corresponding container axis
- the next `AXCTSWE(CTi)` for this axis container

Note

Even an `IC(0)` will result in a wait including synchronization where necessary (block-by-block change in addressing according to increment even though absolute dimension is set globally).

Synchronization with axis position

If the new container axis assigned to the channel after a container rotation does not have the same absolute machine position as the previous axis, then the container is synchronized with the new position (internal `REORG`).

Note

SD41700 AXCT_SWWIDTH[Container No.] is updated only for new configurations.
If, after incremental rotation of the RTM/MS, the position of a circuit before the initial position is reached, the container can be rotated normally in the **forward** direction to reach the initial position of the container again. The drum or rotary table must however be turned **back** to the original position, so that measuring and supply cables are not twisted.

Home channel of an axis container axis

If more than one channel has access authorization ("reference") to the axis via the machine data, write authorization can be passed to the axis (setpoint input):

MD20070 $MC_AXCONF_MACHAX_USED

This machine data below creates a standard assignment between an axis and a channel:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN

The following machine data is used to define which NCU possesses the axis after power up or is producing the interpolation value.

MD10002 $MN_AXCONF_LOGIC_MACHAX_TAB

Since the axial machine data of link axes are the same on all NCUs, the machine data below is only evaluated if the NCU also has write authorization to the axis (see MD 10002):

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN

Axis replacement

Passing the write authorization to an axis (setpoint input) by means of GET, RELEASE, ..., , ..., works for a container axis in the same way as for a normal axis. Write authorization can only be exchanged between the channels of **one** NCU. Write authorization cannot be passed beyond the boundaries of an NCU.

2.7.1 System variables for axis containers

States of axis containers

The following system variables allow part programs and synchronized actions to access information about the current state of an axis container.

Legend:

r: Read

PP: Part program

SA: Synchronized action

SW: Software version

n: Axis container identifier

Name	Type/SW	Description/values	Index	PP access	SA access
$AC_AXCTSWA[n] (AXis ConTainer SWitch Active)	BOOLEAN /5	**Channel** state of axis container rotation/ 1: The channel has enabled axis container rotation for axis container n and this is not yet finished. 0: Axis container rotation is finished. Examples: See Section "Axis container coordination"	Identifier	r	r
$AN_AXCTSWA[n]	BOOLEAN /5	**Axis container rotation**/ 1: Axis container rotation is currently being performed on axis container n. 0: No axis container rotation is active. Examples: See Section "Axis container coordination"	Identifier	r	r
$AN_AXCTAS[n] (AXis ConTainer Actual State)	INT /5	**Current rotation of axis container** This specifies for axis container n by how many slots the axis container has currently been moved./ 0 to (max. number of assigned slots in axis container - 1) The original assignment applies after Power on. This is the value 0.	Identifier	r	r

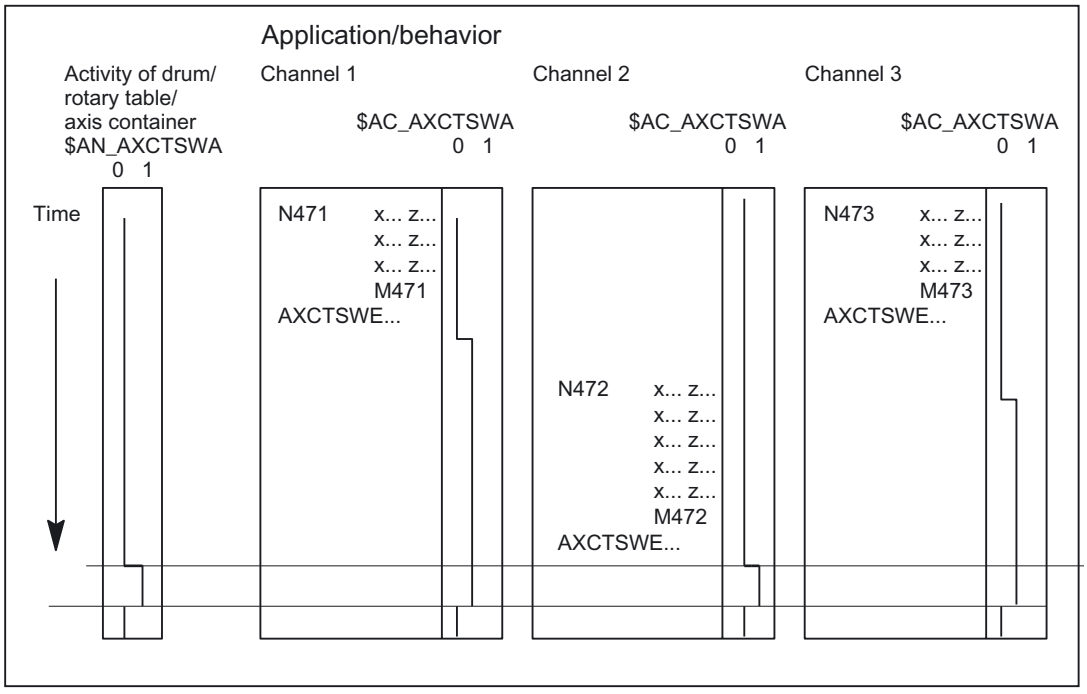


Figure 2-22 Axis container rotation dependent on enable by channels concerned

2.7.2 Machining with axis container (schematic)

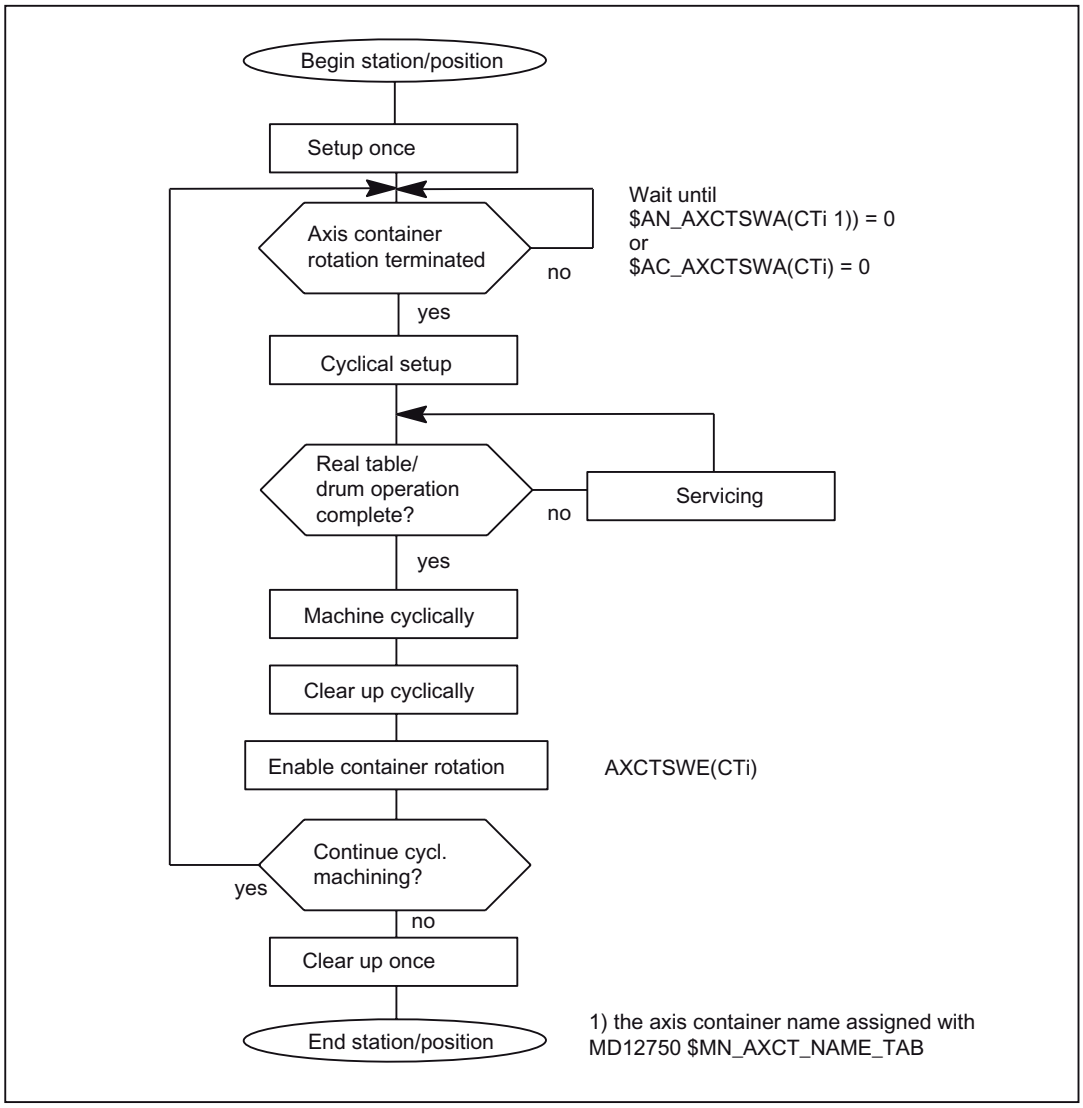


Figure 2-23 Schematic machining of a station/position

Note

An NCU machining cycle which is in charge of the rotation of the rotary table or the drum for multi-spindle machines contains the query of enables for container rotation of all NCUs concerned. If all enables are present, switching to the next position/station takes place. The axis containers are rotated accordingly.

2.7.3 Axis container behavior after Power ON

The container always assumes the state defined in the machine data on `Power On`, irrespective of its status when the power supply was switched off, i.e. the user must distinguish between the actual status of the machine and the default setting and compensate accordingly by specifying appropriate axis container rotations. He can do this, for example, by means of an ASUB containing `AXCTSWED` in one channel while the other channels are still in the `RESET` state.

2.7.4 Axis container response to mode switchover

A container axis in an axis container which has been enabled for rotation cannot be traversed in JOG mode. An axis container can only be rotated in JOG mode by means of an ASUB.

2.7.5 Axis container behavior in relation to ASUBs

An enabling command for axis container rotation cannot be canceled, i.e. if an axis container rotation has been enabled in an ASUB, the enabling command remains effective even when the ASUB has ended.

2.7.6 Axis container response to RESET

A reset cancels the enabling command for axis container rotation. The reset channel is then no longer involved in the axis container rotation. The enabling commands in the other active channels can effect a rotation. If all channels except one have been reset, the one remaining active channel can set the rotary position directly with `AXCTSWED`.

2.7.7 Axis container response to block searches

An axis container rotation (`AXCTSWE`) cannot be enabled and activated in one block, but the enabling and activation commands must be programmed in separate action blocks. In other words, the axis container status changes in response to each separate rotation command as a function of the status of other channels.

2.7.8 Supplementary conditions for axis container rotations

Note

The user must ensure by means of programming that

- the correct zero offsets are active after a container switch
- and that no transformations are active during the container switch.

Axial machine data

If an axis is assigned to an axis container, then certain axial machine data must be identical for all axes in the axis container as the data are activated. This can be ensured by making a change to this type of machine data effective on all container axes and all NCUs which see the axis concerned. The message: "Caution: This MD will be set for all container axes" is output at the same time.

During power up, all axial machine data of this type are synchronized with the values of the machine axis in **slot 1 of the axis container**. In other words, the relevant machine data are transferred from the machine axis in slot 1 of the axis container to all other container axes. If machine data with other values are overwritten by this process, the message: "The axial MD of the axes in axis container <n> have been adapted" is output.

If a slot in the axis container is re-assigned (through writing of machine data MD12701-12716 AXCT_AXCONF_ASSIGN_TAB<n>), the message: "The MD of the axes in axis container <n> will be adapted on next power up" is output.

Axial machine data of the type discussed above are identified by attribute *containerEqual* (equal for all axes in the axis container). With an NCU link, the axis container is defined **on the master NCU** (see Section "NCU link").

Axis states

If a container axis is active in axis mode or as a positioning spindle (POSA, SPOSA) and its axis container needs to be rotated, then the rotation cannot be executed until the container axis has reached its end position.

A container axis which is active as a **spindle continues to turn** as the axis container rotates.

SPCON (switchover to position control) is attached to the physical spindle, i.e. this status is passed on with the spindle when an axis container rotates. SETMS (master spindle), on the other hand, refers to the channel and remains active in the channel when an axis container rotates.

Continuous-path control mode G64

An axis container rotation interrupts G64 mode in a channel in which a container axis in the rotating container is also a channel axis, even if it does not belong to the path grouping. This interruption does not occur, however, until an axis in the rotated axis container is programmed again.

PLC axes

If a container axis in a container which is enabled for rotation must become a PLC axis, then this status change request is stored, and the changeover to PLC axis status does not take place until **after** completion of the axis container **rotation**.

Command axes

A container axis in a container enabled for rotation cannot be declared a command axis. The traverse request is stored in the channel and executed on completion of the axis container rotation.

Exceptions to this rule are synchronized actions M3, M4, M5 and a motion-changing S function: If an axis container rotation is active and the spindle is transferred to the control of another NCU, alarm 20142 (channel %1 command axis %2: Invalid axis type) is output. These synchronized actions do not change a channel axis into a command axis, but leave it in its original state. Synchronized actions of this type cannot be stored.

References: /FBSY/ Function Manual, Synchronized Actions

Reciprocating axes

A container axis in a container enabled for rotation cannot become a reciprocating axis, i.e. this change in status does not take place until the axis container has finished rotating. The status change command remains active.

Axis couplings

An axis container cannot rotate while an axis coupling in which one of its container axes is involved is still active. The coupling must be deselected (`COUPOF`) prior to axis container rotation and selected again (`COUPON`) afterwards. A new `COUPDEF` command is not necessary.

Main run offset values

The main run offset values (DRF offset, online tool offset, synchronized action offset, compile cycle offset) for a channel axis assigned to a container slot remain valid after the relevant axis container has rotated. External zero offsets cannot remain valid after an axis container rotation as these refer to specific machine axes. If an external zero offset is active, the axis container rotation is rejected with alarm 4022.

Axial frame

The axial frame of a channel axis, which is also a container axis, is no longer valid after an axis container rotation. Since the axis container rotation assigns a new machine axis to the channel axis, but the axial frame is referred to a machine axis, **the rotation thus also changes the axial frame**. If the two frames do not coincide, a synchronization process (internal `REORG`) is performed.

The assignment between a channel axis and a machine axis is altered by the axis container rotation. The current frames remain unchanged after a rotation. The user himself is responsible for ensuring that the correct frames are selected after a rotation by programming basic frame masks, for example.

Transformations

If the container axis is a spindle which is involved in a transformation, then the **transformation** must be **deselected** before the axis container rotation is enabled. Otherwise alarm 17605 is activated.

Gantry grouping

Gantry axes cannot be axes in an axis container.

Drive alarms

When a drive alarm is active for a container axis, then the associated axis container cannot rotate until the alarm cause has been eliminated.

2.8 Link communication

2.8.1 Link variables

Function

Complex systems often feature multiple NCUs, each with multiple channels. Each NCU has a link communication channel for the purpose of coordinating manufacturing processes throughout the entire system. Using this, each of the NCUs can exchange data cyclically with every other NCU in the link grouping.

The link communication channel is based on a memory area on each NCU known as the link variables memory. The user/machine manufacturer can define both the size and data structure of the link variables memory on a system-specific basis. The data stored in the link variables memory is addressed by what are known as link variables;

these are system-global user variables which can be read and written in part programs and cycles by all NCUs involved in a link grouping if link communication has been configured. Unlike global user variables (GUD), link variables can also be used in synchronized actions.

On systems without an NCU link, link variables can be used as additional global user variables alongside standard global user variables (GUD).

Requirements

The following requirements must be met in order to use link variables for cross-NCU data exchange:

- The link grouping must be installed and configured. See "Configuration of a link grouping (Page 174)".
- Link communication must be activated:

  MD18780 $MN_MM_NCU_LINK_MASK (activation of NCU link communication)

Properties of the link variables memory

Assigning parameters for the memory size

The size of the link variables memory in bytes is set by means of the following machine data:

MD18700 $MN_MM_SIZEOF_LINKVAR_DATA (size of the NCU link variables memory)

The setting for the size of the link variables memory should be identical for all NCUs involved in the link grouping. If the memory sizes are different, the largest value assigned is used.

Initialization

After an NCU is powered up, the link variables memory is initialized with 0.

Structure

From the point of view of the system, the link variables memory is an unstructured memory area that is available for link communication purposes. The link variables memory is structured by the user/machine manufacturer alone. Depending on how the data structure is defined, the link variables memory is accessed by means of data format-specific link variables.

System-wide alignment

Once a link variables memory has been written to, the changes that have been made to the data are transferred to the link variables memories of all other NCUs involved in the link grouping. The link variables memories are usually updated by means of link communication within two interpolation cycles.

Properties of the link variables

The link variables memory is accessed via the following data format-specific link variables:

Data type 1)	Name	Data format 2)	Bytes 2)	Index i 3)	Value range
UINT	**$A_DLB[i]**	BYTE	1	i = n * 1	0 to 255
INT	**$A_DLW[i]**	WORD	2	i = n * 2	-32,768 to 32,767
INT	**$A_DLD[i]**	DWORD	4	i = n * 4	-2,147,483,648 to 2,147,483,647
REAL	**$A_DLR[i]**	REAL	8	i = n * 8	±(2.2*10^{-308} to 1.8*10^{+308})

1) Data type of link variables when used in part program/cycle

2) Data format of link variables or number of bytes addressed by the link variables in the link variables memory

3) The following must be noted for index i:

- Index i is a byte index that relates to the beginning of the link variables memory.
- The index must be selected so that the bytes addressed in the link variables memory are located on a data format limit ⇒ index i = n * bytes, where n = 0, 1, 2, etc.
  - `$A_DLB[i]`: i = 0, 1, 2, etc.
  - `$A_DLW[i]`: i = 0, 2, 4, etc.
  - `$A_DLD[i]`: i = 0, 4, 8, etc.
  - `$A_DLR[i]`: i = 0, 8, 16, etc.

Writing

A link variable is written with main-run synchronism.

Reading

A preprocessing stop is triggered when a link variable is read.

Checks

The following checks are performed for the link variables and link variables memory:

- Observance of the value range limits
- Access to format limit
- Observance of defined memory area in link variables memory

The user/machine manufacturer is solely responsible for preventing the following errors:

- Accessing with incorrect data format
- Accessing the wrong address (index i)

- Reciprocal overwriting of the same data item by multiple channels of a single NCU or different NCUs
- Reading a data item before it has been updated by a channel of its own NCU or of a different NCU

NOTICE
Data consistency The user/machine manufacturer is solely responsible for ensuring data consistency within the link variables memory, both on a local-NCU basis and across NCUs.

Write elements

In the case of write access to the link variables memory (e.g. $A_DLB[4] = 21), what is known as a link variables write element is required for managing the write process within the system. The maximum number of write elements that are available for each interpolation cycle is set by means of the following machine data:

MD28160 $MC_MM_NUM_LINKVAR_ELEMENTS

The maximum number of write elements thus restricts the number of link variables that can be written during each interpolation cycle.

Dynamic response during write

The link variables are written with main-run synchronism. The new value may be read by the other channels in its own NCU no later than the next interpolation cycle. It can be read in the next part program block in its own channel.

The channels of the other NCUs in the link grouping normally see the new value after two interpolation cycles. However, the limited bandwidth can lead to delays in transferring write jobs to the other NCUs in the link grouping (message delays). Possible situations include:

- Writing a large number of link variables in an interpolation cycle
- Writing link variables and requesting an axis container rotation in the same interpolation cycle
- Writing link variables and transferring an alarm in the same interpolation cycle

Synchronizing a write job

If certain applications require the new value of a link variable to be transferred to the other NCUs in the link grouping in precisely two interpolation cycles, writing may be carried out by means of a synchronized action, and the $A_LINK_TRANS_RATE system variable written in accordance with the number of write elements available. The system variable contains the value specifying how many write jobs can still be transferred to the other NCUs in the link grouping in the current interpolation cycle.

Programming example

```
N120 WHEN $A_LINK_TRANS_RATE > 0 DO $A_DLB[0] = 9
N125 WHEN $A_LINK_TRANS_RATE > 0 DO $A_DLB[1] = 7
```

The byte addressed by link variable `$A_DLB[0]` is only written to `N120` if the write job can be transferred to the other NCUs in the link grouping in the same interpolation cycle. If this is the case, the link variable is written and the system variable decremented at the same time. A block change then takes place and `N125` is processed in the same way.

If it is not possible for the write job to be executed immediately in `N120`, an attempt is made to execute the write process in the next interpolation cycle.

Example

The following data is required for link communication:

Data format	Number	Bytes required
BYTE	2	2
WORD	1	2
DWORD	3	12
REAL	1	8
		24 (required size of the link variables memory)

The data is arranged in the link variables memory as follows, with the data format limits taken into account:

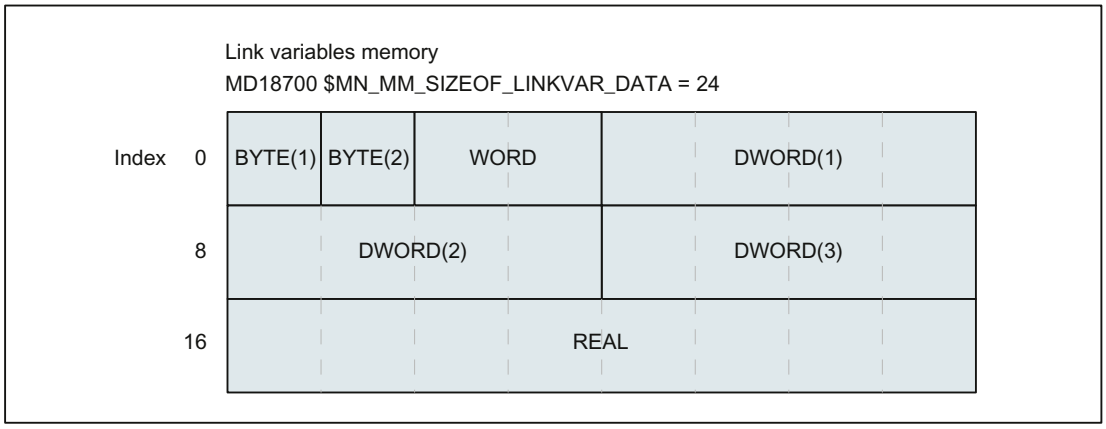


Figure 2-24 Example: Structure of the link variables memory

Note

Memory structure

The data in the link variables memory is always arranged randomly and may therefore appear different (although the data format limits will still be taken into account).

Access via a link variable must be programmed as follows, in accordance with the memory structure defined:

Program code	Description
$A_DLB[0]	; BYTE(1)
$A_DLB[1]	; BYTE(2)
$A_DLW[2]	; WORD
$A_DLD[4]	; DWORD(1)
$A_DLD[8]	; DWORD(2)
$A_DLD[12]	; DWORD(3)
$A_DLR[16]	; REAL

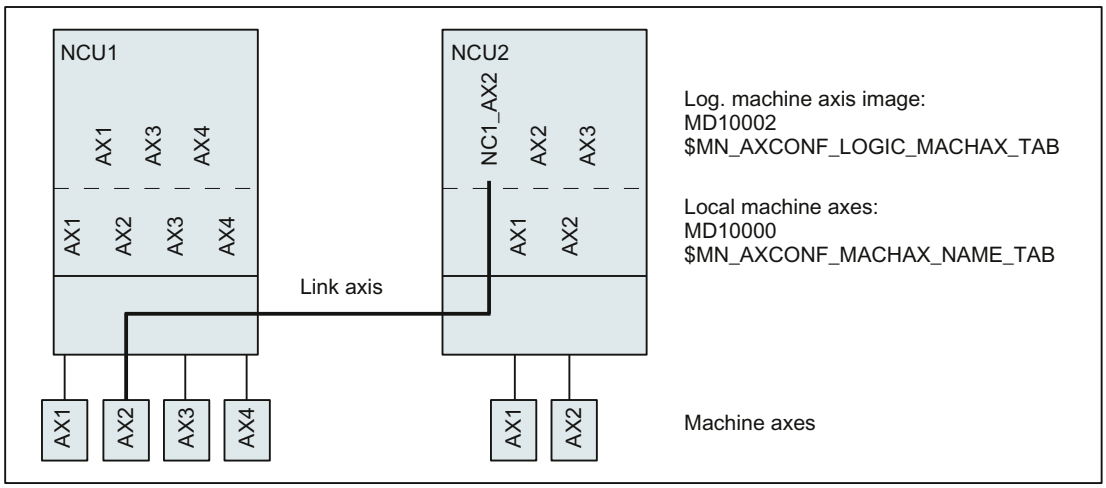
See also

Reading drive data via link variables (Page 172)

2.8.2 Reading drive data via link variables

Task

A system contains 2 NCUs called NCU1 and NCU2. The two NCUs are connected via an NCU link. Several machine axes are connected to NCU1. Of these, axis AX2 is traversed in an interpolatory manner as a link axis of NCU2. The actual current value of axis AX2 is to be transferred via link communication from NCU1 to NCU2 for evaluation purposes. The figure below shows the basic design of the system.



Requirements

The actual current value of axis AX2 can be read via system variable $VA_CURR. In the case of PROFIdrive-based drives, the following machine data needs to be set for this purpose:

MD36730 $MA_DRIVE_SIGNAL_TRACKING = 1 (acquisition of additional drive actual values)

Setting the machine data makes the following drive actual values available:

- $AA_LOAD, $VA_LOAD (drive capacity utilization in %)
- $AA_POWER, $VA_POWER (drive active power in W)
- $AA_TORQUE, $VA_TORQUE (drive torque setpoint in Nm)
- $AA_CURR, $VA_CURR (actual current value of axis or spindle in A)

Programming

NCU1

A static synchronized action is used to write actual current value $VA_CURR of axis AX2 to the first 8 bytes of the link variables memory cyclically in the interpolation cycle, via link variable $A_DLR[0] (REAL value):

Program code

```
N111 IDS=1 WHENEVER TRUE DO $A_DLR[0]=$VA_CURR[AX2]
```

NCU2

A static synchronized action is used to read actual current value $VA_CURR of axis AX2, transferred via link communication, cyclically in the interpolation cycle via link variable $A_DLR[0]. If the actual current value is greater than 23 A, alarm 61000 is displayed.

Program code

```
N222 IDS=1 WHEN $A_DLR[0] > 23.0 DO SETAL(61000)
```

2.9 Configuration of a link grouping

Introduction

The preceding sections described how to configure link axes and axis containers. Both require a link communication to be established between the NCUs concerned. Setting up the link communication takes place by means of:

- The link module hardware
- **References:** /PHD/ SINUMERIK 840D Configuration Manual
- Machine data

The following section describes how to use the required machine data.

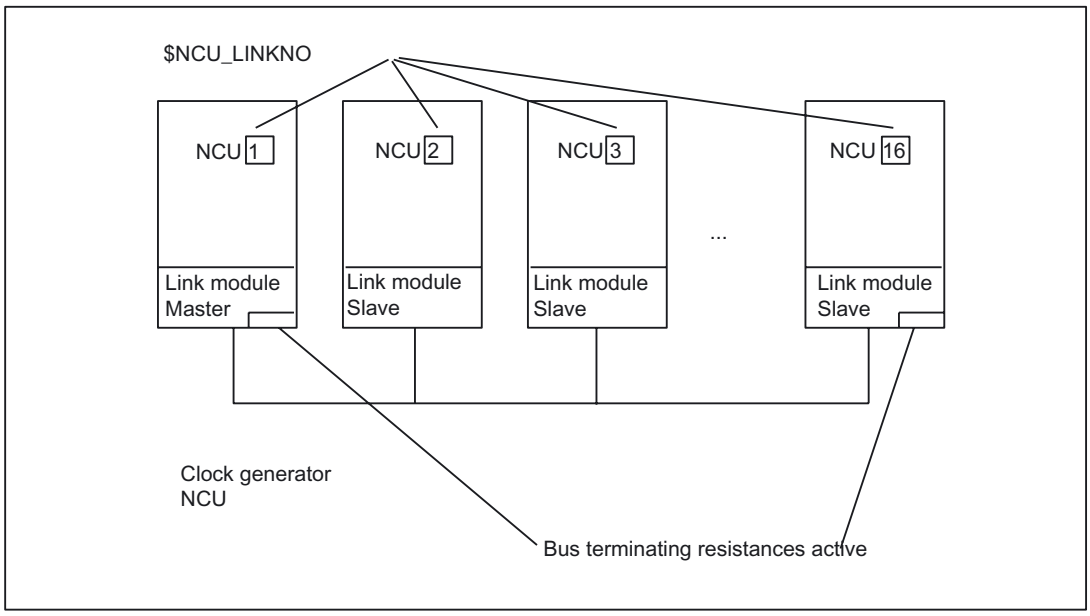


Figure 2-25 Link grouping

Link grouping

A link grouping consists of a minimum of 2 and maximum of 16 NCUs interconnected by link modules.

- Link module master

  (MD12510 $MN_NCU_LINKNO = 1)

  plays a leading role in this process. It synchronizes the interpolation cycle and sets up slave communication on power up. It is advisable to assign the NCU numbers in continuous ascending order for the slave modules.
- The first and last module in the physical chain must activate the bus terminating resistors.
- The software version must be identical on all NCUs in a link grouping.

Machine data

MD18780 $MN_MM_NCU_LINK_MASK

This machine data ensures that link communication is established. It provides the dynamic memory space that is required for communication in the NCUs equipped with link modules.

The machine data below determines the data transfer rate of link communication based on the assignments listed below:

MD12540 $MN_LINK_BAUDRATE_SWITCH

Set value/Rate:

0: 9,600 Kbaud

1: 19,200 Kbaud

2: 45,450 Kbaud

3: 93,750 Kbaud

4: 187,500 Kbaud

5: 500,000 Kbaud

6: 1,500 Mbaud

7: 3,000 Mbaud

8: 6,000 Mbaud

9: 12,000 Mbaud (default setting)

Machine data:

MD12550 $MN_LINK_RETRY_CTR

specifies the maximum number of times the link communication is automatically repeated when an error occurs during frame transfer.

The machine data below specifies the number of link modules taking part in the link communication:

MD12530 $MN_LINK_NUM_OF_MODULS

The following machine data assigns a logical link number to an NCU; this number is used for link identification in conjunction with link axes and link communication:

MD12510 $MN_NCU_LINKNO

Identifications can be assigned independently of the physical sequence of the modules in the link string. The module with NCU_LINKNO = 1 is **master**.

⚠ WARNING
Assignment of NCU_LINKNO must be unambiguous. An alarm is issued if there is an error.

The machine data below specifies for the software which NCUs correspond to the bus terminating resistors:

MD12520 $MN_LINK_TERMINATION

The set values refer to the entries defined with machine data:

MD12510 $MN_NCU_LINKNO

0 corresponds to the first definition, 1 to the second, etc. from machine data:

MD12510 $MN_NCU_LINKNO.

Note

The following machine data need only be set for the prototype hardware of the link module:

MD12520 $MN_LINK_TERMINATION

It is only meaningful for the software.

The NCUs that are **physically** connected at the **beginning** and **end** of the bus must activate the terminating resistors. This measure is necessary for the link communication to work.

The machine data below defines for the purposes of power up which NCU in an NCU grouping will be responsible for generating the axis setpoint (master NCU):

MD30554 $MA_AXCONF_ASSIGN_MASTER_NCU

The machine data below specifies that the axis drive needs to be started during power-up, even if the axis is operating under the control of another NCU.

MD30560 $MA_IS_LOCAL_LINK_AXIS

It is evaluated only if machine data required to create a link grouping have been set, but link communication has failed due to an error.

Note

It may be necessary to increase the interpolation cycle due to the number of link axes and write elements.

2.10 Communication in link grouping

Although communication by means of link modules is high-speed communication, the following aspects have to be taken into account during configuration.

Data transport

Both cyclic and acyclic services are used for data communication. The cyclic data area ensures that the data are transferred in every interpolation cycle.

The acyclic service uses machine data, link variables and data for container rotation, for example. If too many data were selected simultaneously for the acyclic link communication, the transfer sequence is governed by an internally defined priority system. The actual value/setpoint data for the lead link axis are transferred via the cyclic service.

Dependencies

The times for communication are determined by:

- Number of data for cyclic exchange

  (number of link axes, number of lead link axes (see below), number of axes in a cross-NCU container with high priority and number of link variables having to be exchanged in each interpolation cycle)
- Type and speed of the NCUs
- Link module speed (currently 12 Mbaud)
- Point-to-point protocol: The cyclic communication load increases sharply with each additional NCU, as illustrated by the schematic diagram "Increase in communication time with the number of NCUs connected via the link (for scaling refer to Interdependencies)".

The set interpolation cycle must be identical for all lead link NCUs in the axis grouping.

Link resources

The machine data below controls the assignment of send/receive buffers for link connections:

MD18781 $MN_NCU_LINK_CONNECTIONS

A maximum of 32 buffers of this kind are available for each NCU.

One of these buffers is reserved on a priority basis for the setpoint/actual value transfer of a link axis. Only the remaining ones can be used for acyclic data exchange (alarm, container switch, link variable transfers). The maximum number of 32 must not be exceeded.

If the machine data below is set with 0, the software itself determines the requirement as 25 connections:

MD18781 $MN_NCU_LINK_CONNECTIONS

A value not equal to 0 explicitly defines the number of acyclic connections to other NCUs.

Examples

Let an axis container contain 12 slots. Three axes are local on NCU1, and three link axes are located on each of NCU 2, NCU3, and NCU4. The MD is set with 0 as the value.

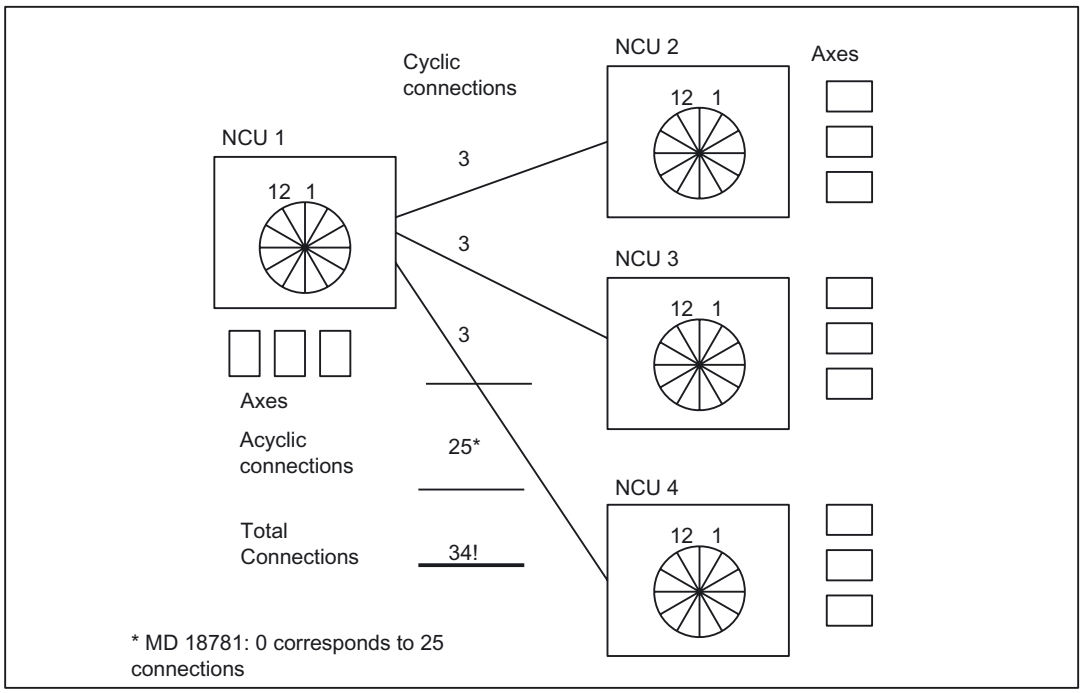


Figure 2-26 Resources insufficient

Let an axis container contain 12 slots. Four axes are local on NCU1, four axes on each of NCU 2 and NCU3 are link axes. The machine data is set with 9 as the value:

MD18781 $MN_NCU_LINK_CONNECTIONS

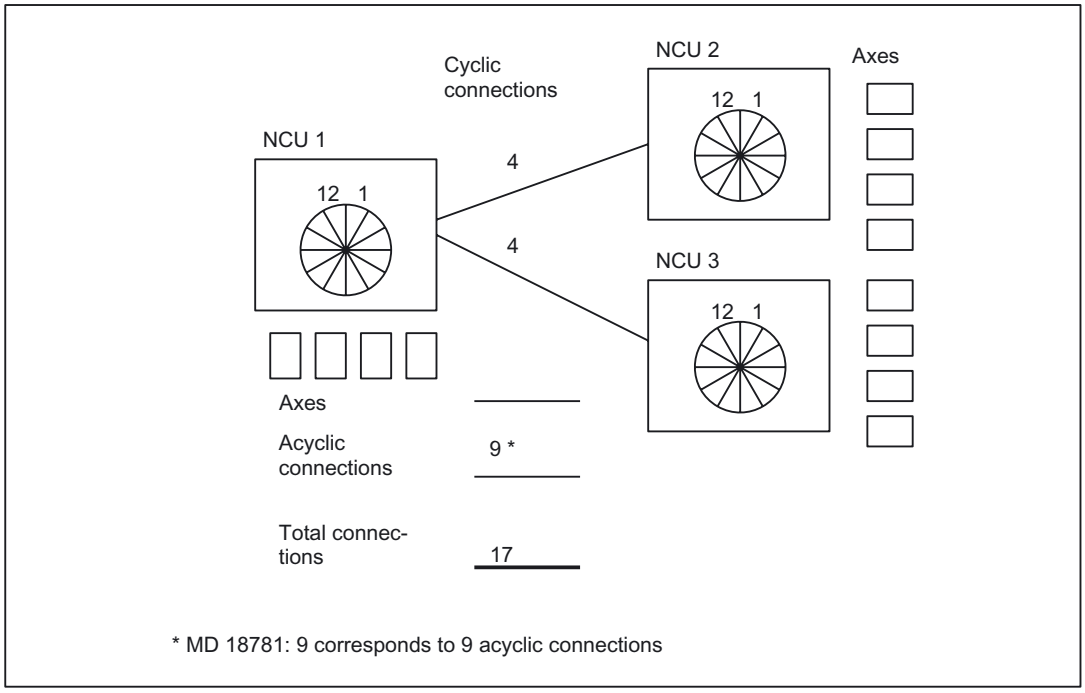


Figure 2-27 Resources sufficient

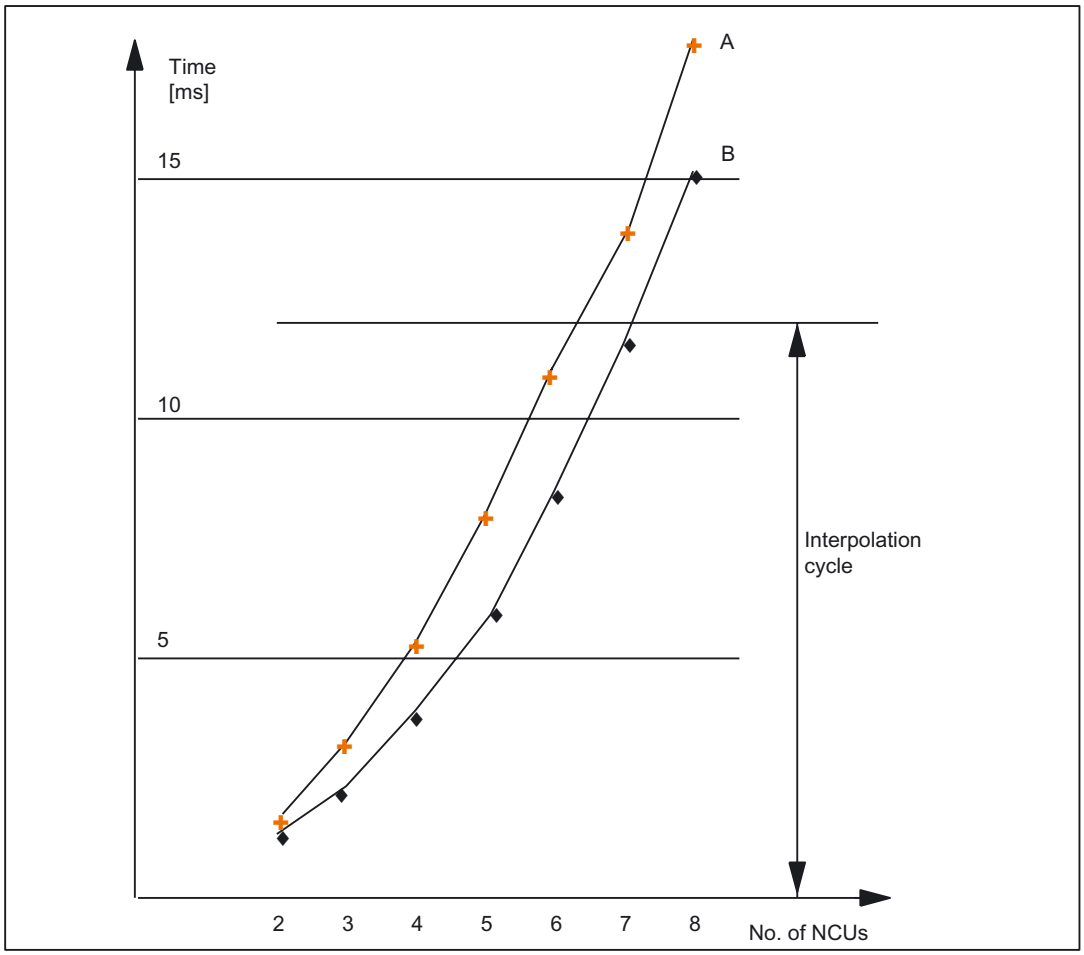


Figure 2-28 Increase in communication time of the number of NCUs connected via the link (for scaling refer to Interdependencies)

Configuration limit

The diagram above illustrates how the communication overhead grows as the number of NCUs increases.

Curve trace A:

Time required for the exchange of link variables/machine data information and the lead link axis information (one lead link axis) between the NCU giving the master value and other NCUs that interpolate the following axes as a function of the leading axis (lead link axis).

Curve trace B:

Time required for the exchange of link variables/machine data information between the NCU giving the master value and other NCUs that interpolate the following axes as a function of the leading axis (lead link axis).

Rule

In a configuration, the time requirement must remain below the interpolation cycle in accordance with curve trace A. If it is not possible to alter the number of required NCUs, the interpolation may have to be adapted if necessary.

If the interpolation cycle has to remain unchanged, the number of NCUs in the lead link axis grouping may have to be reduced if necessary.

2.11 Lead link axis

Term

A lead link axis allows read access to the axis data (setpoint, actual value, ...) on another NCU.

Introduction

The lead link axis concept offers a solution for the following problems:

The individual machining and handling stations are to move synchronous with or in relation to a common master value in so-called clocked sequences. The dependent axes are interpolated from another NCU, not the one interpolating the leading axis.

Characteristics of a lead link axis

Cross-NCU couplings possible.

Several NCUs can be linked to a leading axis of another NCU.

The link grouping can include up to 8 NCUs connected via link communication. Axis values and other information about link communication can be exchanged between the leading axis and the following axes.

The axis or spindle are usually equal in their role as leading unit of an axis coupling.

The same interpolation cycle must be set for all NCUs participating in the coupling.

Restrictions

- An axis (leading axis) that is referenced by a lead link axis cannot be a link axis, i.e. be moved by any other NCUs apart from its home NCU. See "Link axes".
- An axis (leading axis) that is referenced as a lead link axis cannot be a container axis, i.e. addressed alternately by different NCUs. See "Axis container".
- A lead link axis must not itself be a container axis.
- A lead link axis cannot be the programmed leading axis in a gantry grouping.
- Couplings with lead link axes cannot be cascaded.
- Axis replacement can only be implemented within the home NCU of the lead link axis. See below.

Lead link axes allow cross-NCU couplings where the master value axis and the following axes are programmed/interpolated on **different NCUs** (see supplementary conditions).

Couplings

The following coupling types can be used:

- Master value (setpoint, actual value, simulated master value)
- Coupled motion
- Tangential correction
- Electronic gear (ELG)
- Synchronous spindle

Configuration of leading axis NCU

The lead link axis that is being interpolated as a leading axis on the NCU is configured on the interpolating NCU as a standard local axis. It is configured by means of channel axis and machine axis machine data, and via the logical machine axis image. In addition, there are the following requirements:

- The machine axis must be identified as lead link axis in machine data:

  MD30554 $MA_AXCONF_ASSIGN_MASTER_NCU
- Number of link module:

  MD12510 $MN_NCU_LINKNO
- Activation of link functionality:

  MD18780 $MN_MM_NCU_LINK_MASK = 1
- Number of link modules:

  MD18782 $MN_MM_LINK_NUM_OF_MODULES
- Size of servo buffer specified in:

  MD18720 $MN_MM_SERVO_FIFO_SIZE = **4**

Configuration of following axis NCU

In addition to configuring the standard channel/machine axis machine data, you need to configure machine axis machine data on the NCU that is deriving following axis movements from the non-local leading axis (lead link axis):

- Number of link module:

  MD12510 $MN_NCU_LINKNO
- Activation of link functionality:

  MD18780 $MN_MM_NCU_LINK_MASK = 1
- Number of link modules:

  MD18782 $MN_MM_LINK_NUM_OF_MODULES
- Possibly MD18402 $MN_MM_NUM_CURVE_SEGMENTS
- Possibly MD18404 $MN_MM_NUM_OF_CURVE_POLYNOMS
- Size of servo buffer specified in:

  MD18720 $MN_MM_SERVO_FIFO_SIZE = **2**

- The lead link axis must be configured in the logical machine axis image with machine data:

  MD AXCONF_LOGIC_MACHAX_TAB[i] = "NC_m_AX_n"

  This allows a relation to be established with the NCU that is interpolating the lead link axis.
- The lead link axis must also be configured in the channel axes.

Note

For further information about link-specific machine data, please refer to the preceding subsections and the configuration and programming example provided in Section "Examples".

Schematic sequence

The diagram below shows the constellation used in the example with the main data flows.

NCU 1 provides:

Interpolation and position control for master value axis or supplies the lead link axis of NCU2 with setpoint and actual value.

NCU 2 provides:

Coupling between lead link axis and master value axis (of NCU1) and generation of a dependent axis motion (following axis).

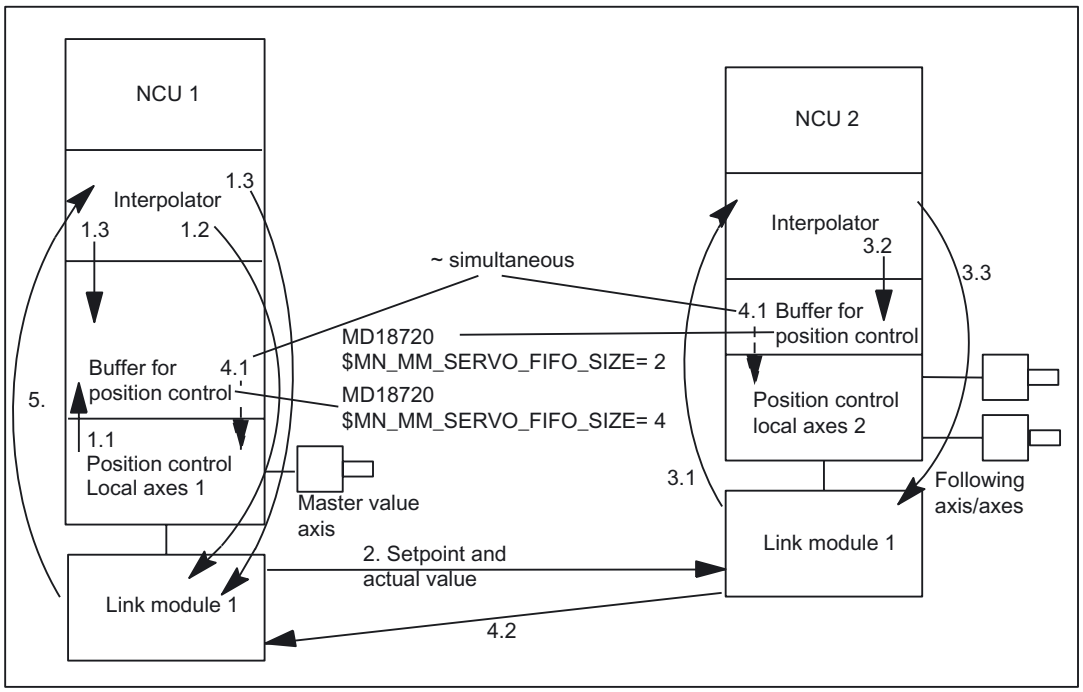


Figure 2-29 Data flows for leading value axis, lead link axis and subordinate following axis/axes

The following steps are illustrated:

- 1.1 Position control on NCU1 reads in actual values of leading value axis from the drive and writes them in the communication buffer for interpolation.
- 1.2 In the same cycle as (1.1), the actual values from the NCU1 interpolator are written to the link module.
- 1.3 NCU1 interpolator writes the generated setpoints of the leading value axis to the link module and in the local buffer for position control.
- 2. Setpoints and actual values are transferred from NCU1 to NCU2 via the link module.
- 3.1 NCU2 interpolator receives setpoints of the leading value axis via the link module and calculates the setpoints of the following axis.
- 3.2 NCU2 interpolator writes the generated setpoints of the following axis in the buffer for position control.
- 3.3 NCU2 interpolator sends reaction feedback of following axis to leading value axis to the link module.
- 4.1 NCU1 position control receives setpoint for leading axis (delay caused by: MM_SERVO_FIFO_SIZE=4). NCU2 position control receives setpoint for the following axis.
- 4.2 Reaction feedback is transferred from NCU2 to NCU1.
- 5. NCU1 interpolator receives reaction feedback and can incorporate it when calculating setpoints for the leading value axis.

Transporting lead link data

The system variables $AA_LEAD_SP and $AA_LEAD_SV (see below) are transferred via the acyclic service. These system variables have a lower transfer priority than the link variables.

Approx. 320 bytes of data are exchanged for the lead link axis (setpoints and actual values). The communication time required for this operation is approx. equal to that required for a link axis.

In contrast to local NCU coupling, the leading axis setpoints (on NCU1) with cross-NCU couplings are delayed to achieve synchronously clocked assignments of setpoints for the leading/following axis. (See step 4.1)

The set interpolation cycle must be identical for all lead link NCUs in the axis grouping.

Axis replacement

The channels of the NCU that is driving the following axes are not allowed to traverse or replace a lead link axis. The real leading axis can be replaced on its home NCU.

The commands GET, GETD and also Auto-Get ($MA_AUTO_GET_TYPE) are rejected by the alarm "Channel %1 axis replacement for axis %1 not allowed".

2.11.1 Programming a lead link axis

Master value axis view

Only the NCU which is physically assigned to the master value axis can program travel motions for this axis. The travel program must not contain any special functions or operations. The configuration summarized in the preceding sections allows the control to perform the required setpoint delay automatically so that the leading axis and following axes are interpolated synchronously on the other NCUs.

Following axes view

The programming on the NCU of the following axes must not contain motion commands for the leading value axis (master value axis). Any violation of this rule triggers an alarm.

The leading value axis is addressed in the usual manner via the channel axis identifier. The states of the leading value axis can be accessed via special system variables.

System variables

The following system variables can be used in conjunction with the channel axis identifier of the master value axis:

$AA_LEAD_SP; Simulated master value position

SAA_LEAD_SV; Simulated master value velocity

If these system variables are updated by the home NCU of the master axis, the new values are also transferred to any other NCUs which wish to control slave axes as a function of this master axis.

2.12 NCU link with different interpolation cycles

Problem description

In the engineering world, parts which deviate slightly from a precise round/cylindrical shape are also required. (Example: Pistons that are oval in the manufacturing state. The operating temperature gives them their required almost round shape during use). Eccentric shapes include, e.g.:

- Oval shapes
- Trefoil shapes
- Cams
- More complex shapes

If you want to manufacture parts belonging to this category cost-effectively on turning machines, you can use the link functions described below, which are an extension of the link functions described in "NCU link" and the following sections. The supplementary conditions must be complied with, however.

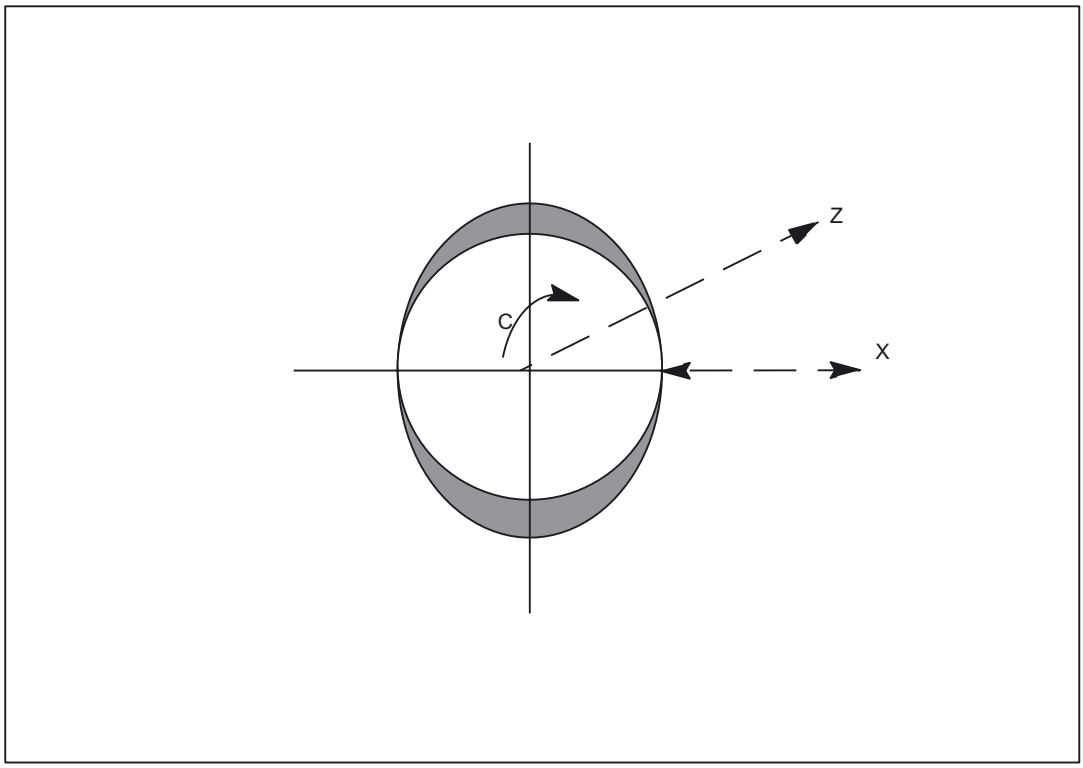


Figure 2-30 Example of oval eccentric turning

Motion sequences

While the workpiece is rotating about the C axis, the X axis must be advanced with high precision between the smallest and largest radius/diameter according to the required shape (sine, double sine, etc.). In general, a smaller linear movement per rotation is sufficient in Z direction. If the workpiece is to be moved at a typical speed (e.g. 3000 rpm), the X axis must be highly dynamic. This means:

- shorter interpolation cycle
- shorter position control cycle

compared to the requirements of the C and Z axis.

Solution

The requirements described in the motion sequences are met by an NCU link grouping; where the NCU with the shorter interpolation cycle drives the highly dynamic X axis and another NCU with standard interpolation cycle drives the less dynamic C and Z axes. General polynomials are used to specify the motion for the X axis.

Note

The NCU constellation with different interpolation cycles - where one is always faster than the others - is called "FAST-IPO-LINK" in alarm messages.

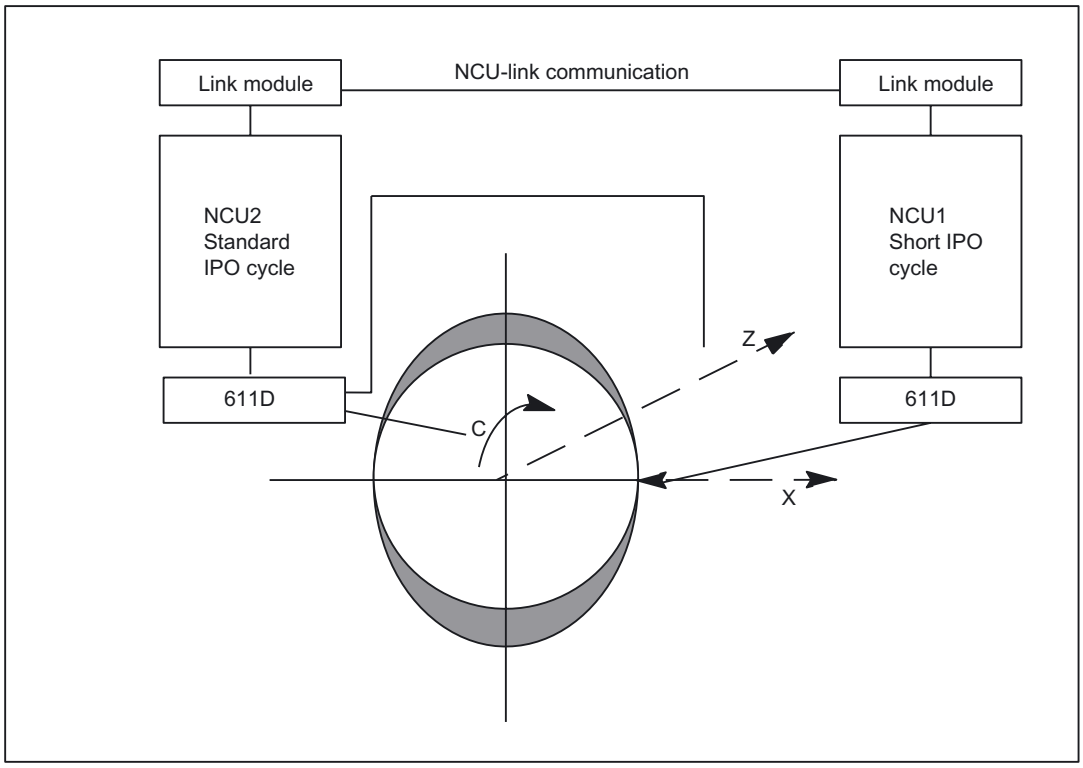


Figure 2-31 NCU link with different interpolation cycles

Generalized solution

In a link grouping with several (up to 8) NCUs, some NCUs are set up with short interpolation cycles, some with standard interpolation cycles, and the axes are configured as in the diagram above. The requirements for NCUs can be optimized in this manner. Eccentric machining is possible for multi-spindle turning machines in conjunction with the axis container concept (see "Axis container").

Note

There is a second method of specifying the motion of the X axis:

by means of master value coupling of the X axis to the spindle (LEADON). This alternative is still possible and is sufficient for tasks with low dynamic performance requirements. However, it does not offer a specific function extension for eccentric turning, and does not allow setpoints to be specified on an NCU with a faster IPO cycle via an NCU link.

References: /PGA/, Programming Manual Job Planning

2.12.1 Diagram of general solution

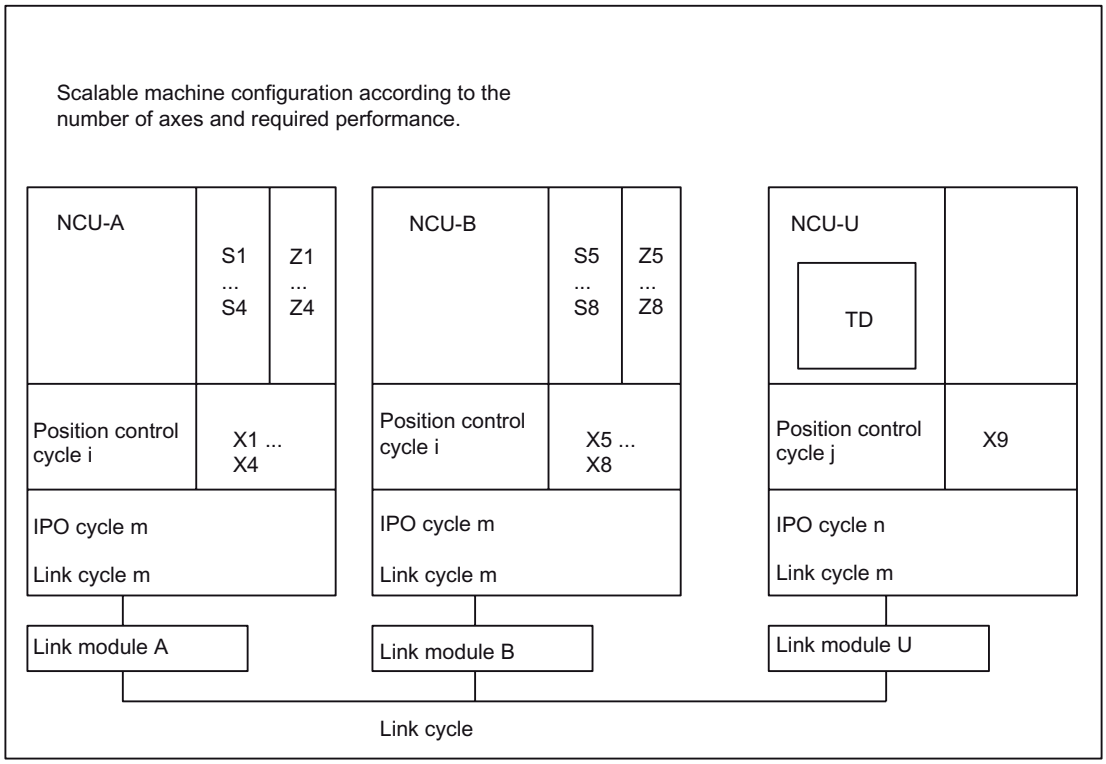


Figure 2-32 Schematic example of a configuration with several NCUs and eccentric machining unit

Abbreviations and terms

NCU-A, NCU-B	NCUs with standard interpolation cycle
NCU-U	Eccentric NCU with fast interpolation cycle
Position control cycle i	"slower" position control cycle
Position control cycle j	"faster" position control cycle
IPO cycle m	"slower" interpolation cycle
IPO cycle n	"faster" interpolation cycle
Link cycle m	Uniform link cycle from NCU-A to NCU-U
S1-S8	Spindles 1-8
Z1-Z8	Z axes 1-8
X1-X8	X axes for concentric machining via NCU-A/NCU-B
X9	X axis for eccentric machining
TD	On NCU-U channels are defined which drive the spindles and Z axes of NCU-A/NCU-B as link/container axes and interpolate the highly dynamic X axis with fast interpolation and position control cycles.

Cycles

The IPO cycle m is an integer multiple of IPO cycle n.

Position control cycle i may be greater than fast IPO cycle n.

Slow IPO cycle m is used as a communication cycle between NCU-A/NCU-B and NCU-U.

Spindle speed

The part program manufacturer must first perform an `OFFLINE` calculation to ascertain the maximum spindle speed to program so that the axes (especially the X axis) are not overloaded when accelerating. If this maximum speed is incorrect, the spindle speed is reduced at points where one of the participating axes would be overloaded. This in turn has a bearing on contour precision and should therefore be avoided.

Contour deviations

Spindle decelerations/accelerations lead to deviations from the programmed contour because the cycles of NCU-A/NCU-B and NCU-U have different lengths; therefore, the acceleration processes should already be terminated before initiating eccentric contour machining.

Axis assignments

Link axes may only be connected to NCUs that have the same IPO cycle as the link cycle.

Example: Axis X9 on the fast NCU-U in the diagram above cannot be interpolated as a link axis by NCU-A or NCU-B. However, axis Z4 can be interpolated by NCU-A, NCU-B and NCU-U.

Setpoint delay

From NCU-U's point of view, it is interpolating with the local X axis and the two link axes C and Z when performing eccentric machining. To achieve a correct contour, it is necessary to compensate for the time delay which occurs when the setpoints are transferred to the link axes and for the differences in clock cycles by delaying the setpoints to the local X axis. This can be done by setting machine data:

MD18720 $MN_MM_SERVO_FIFO_SIZE.

Use the following formula for assigning MM_SERVO_FIFO_SIZE:

MM_SERVO_FIFO_SIZE = 2 * link cycle / IPO cycle +1

The formula applies for NCU-U and NCU-A/NCU-B as illustrated in the diagram "Schematic example of a configuration with several NCUs and eccentric machining unit" (Section "Diagram of general solution").

Examples:

NCU-U:

Link cycle = 16 msec

IPO cycle n = 2 msec

The cycle ratio is therefore 8; the value of the formula 2 * 8 + 1 = 17

NCU-A/NCU-B:

Link cycle = 16 msec

IPO cycle m = 16 msec

The cycle ratio is therefore 1; the value of the formula 2 * 1 + 1 = 3

In some cases, incorrect settings cause an alarm to be output in machine data:

MD18720 $MN_MM_SERVO_FIFO_SIZE

Effect of the SERVO_FIFO_SIZE settings

- All functions that use actual values when interpolating for setpoint generation will be affected by the delay of the slower link cycle instead of the fast IPO cycle. This also applies to the response at faults (alarms that are issued to disable a mode group or for interpolatory braking).
- Alarm responses that are triggered directly in the servo are not affected by any additional delay.

- Acceleration processes on the link axes are always output with the link cycle and are therefore not exactly synchronous with the axes physically connected to the fast NCU. Therefore, it is only advisable to use this configuration with interpolating axes on NCUs with different IPO cycles if you are machining with link axes that are only accelerated a little or not at all. Thus the axes with high dynamic requirements must be physically connected to the fast NCU.
- If high precision is required for the machining operation, the tool should not have contact with the workpiece during acceleration/deceleration of the axes. Otherwise contour violations will result.
- It is only permissible to switch axis containers with **rotating** spindles on NCUs with IPO cycles that are equal to the link cycle. If it is possible to delay the container rotation, switching is delayed until the spindle is motionless. Otherwise alarm 4019 is issued. Axis container switching with **motionless** axes and spindles is also permissible on the faster NCUs. (See "Axis container").

2.12.2 Different position control cycles

The general solution described in Section "Diagram of general solution" also permits different position control cycles for NCU-A/NCU-B and NCU-U as illustrated in the diagram "Schematic example of a configuration with several NCUs and eccentric machining unit" (Section "Diagram of general solution").

By setting the parameters as described below, you can improve the quality for some controller types.

When interpolating, the different cycle times are detected in fine interpolation and in the position control due to the different position control cycles; they are compensated for internally by delaying the setpoints for the axes with the faster position control cycle.

Standard result

If the axis with the slower position control cycle is not accelerated or is only accelerated very slowly, contour errors are largely avoided. However, contour errors will always occur when accelerating, therefore accelerations should not take place while machining.

Special MD settings

In some cases, it may be necessary to be able to adapt the delay (see examples below). This machine data was introduced for this purpose:

MD10065 $MN_POSCTRL_DESVAL_DELAY

It allows you to adapt the setpoint delay in the position controller **for the entire NCU**. You can use positive and negative values. The set values (time in seconds, max. +/- 0.1sec) are added to the automatically calculated values. See under "Information" below.

Setting notes

The required position setpoint delay depends on the controller structure used (DSC (dynamic stiffness control), feedforward control); this delay is taken into account when switching to the respective controller (e.g. FFWON (travel with feedforward control ON)). The **automatic** setting function obtains the maximum value for the position control cycles of the individual NCUs and delays the times on all the NCUs, thus providing compensation at constant speed for the different cycle times on all types of controller structures. In this manner, an additional delay can even be active on the axis with the slowest position control cycle.

The controller structure with

- the highest compensation requirement is operation of torque feedforward control or with speed feedforward control and activated ramp encoder in the drive,
- the least compensation requirement is DSC or if feedforward control is deactivated.

Information

The setpoint delay for the three controller structures

- Without feedforward control (index 0)
- Speed feedforward control (index 1)
- Torque feedforward control (index 2)

is displayed in read-only machine data:

MD32990 $MA_POSCTRL_DESVAL_DELAY_INFO

Negative values in machine data:

MD10065 $MN_POSCTRL_DESVAL_DELAY

reduce the values for **all three** controller structures; positive values increase them.

Appropriate changes to parameter settings

A change in the parameter settings of machine data:

MD10065 $MN_POSCTRL_DESVAL_DELAY

is appropriate in the following cases:

- DSC is always activated or feedforward control is never activated for axis operation.

  In these cases, an additional delay is only necessary on the NCU with the fast position control cycle. Here the following procedure is appropriate:

  1.) MD32990 $MA_POSCTRL_DESVAL_DELAY_INFO[0] on any axis on the NCU with the slowest position controller cycle.

  2.) Subtract this time from the current value in machine data:

  MD10065 $MN_POSCTRL_DESVAL_DELAY

  on all NCUs.

- With the typically used controller structure, there will be a delay in machine data:

  MD32990 $MA_POSCTRL_DESVAL_DELAY_INFO

  in the axis with the fast position control cycle; the delay is not an integer multiple of the position control cycle.

  As there are particularly high dynamic performance requirements on the axis with the fast position control cycle, even the necessary averaging for generating the delay may lead to problems.

  Example: Typically, machining is conducted with speed feedforward control. In axis X of NCU-U in the diagram "Schematic example of a configuration with several NCUs and eccentric machining unit" (Section "Diagram of general solution"), a value of 7 ms is read out in machine data:

  MD32990 $MA_POSCTRL_DESVAL_DELAY_INFO[1]

  The position control cycle on this NCU is 2 ms.

  In this case it may be useful to increase the machine data below on all NCUs by 1 ms:

  MD10065 $MN_POSCTRL_DESVAL_DELAY

  The resulting value of 8 ms in MD32990 POSCTRL_DESVAL_DELAY_INFO[1] will then be an **integer multiple** of the position control cycle of 2 ms.

2.12.3 Supplementary conditions

- The option "different interpolation cycle" can only be used in conjunction with NCU link (options, dependent on the axis number). The connected NCUs must all be fitted with the link module hardware components.
  - The configuration rules outlined in "Configuration of a link grouping" apply.
  - Link axes must be configured as described in Section "Configuration of link axes".
  - Axis containers can be used as described in Section "Axis container".
- It is only permissible to use link axes on NCUs whose IPO cycle length is the same as that of the link cycle,

- Axis container
  - It is only permissible to switch axis containers with rotating spindles on NCUs with IPO cycles that are equal to the link cycle.
  - Axis containers can also be switched with motionless axes and spindle on the NCUs with faster IPO cycles.
- Acceleration and deceleration processes on axes with slower interpolation or position controller cycles will cause contour errors.

2.12.4 Activating NCU links with different interpolation cycles

- Setting different IPO cycle times in one link grouping
- Set bit 1 in machine data:

  MD18780 $MN_MM_NCU_LINK_MASK
- The option "NCU link with different interpolation cycles" is available.

Example

Please refer to Section "Examples" for an example of eccentric turning using the "NCU link with different interpolation cycles" function.

2.12.5 Different IPO cycles, behavior at power ON, RESET, etc.

The necessary setpoint delay on the servo causes a delay for two link cycles during power up, at reset, at New Config, etc. At reset, an additional delay of at least two link cycles is required to account for the necessary synchronization.

2.12.6 System variable with different interpolation cycles

There are no new system variables.

The existing general system variable $A_LINK_TRANS_RATE only displays a value not equal to zero in the **link communication cycle** on an NCU with an IPO cycle with a shorter length than the link communication cycle. The displayed value is available to the user; the transfer is only really conducted in this cycle.

If it is not possible to transfer all the link variables in one cycle, alarm 14764 "NCU link cannot transfer all link variables at once" is displayed as previously. This alarm can be suppressed.

2.13 Link grouping system of units

Introduction

Cross-NCU interpolations are possible in the link grouping with:

- Link axes (see "Link axes")
- Lead link axes (see "Lead link axes")
- Link axes on NCUs with different interpolation cycles (see "NCU link with different interpolation cycles")

With all constellations, it is only possible to achieve a correct result if all the NCUs connected via the NCU link are using the same system of units.

Link grouping, global setting

To ensure consistency across the configuration, the **operator input** switchover of system of units (metric/inch) on the operator panel of one NCU affects the entire NCU grouping with software version 5 and higher.

If the conditions:

- MD10260 $MN_CONVERT_SCALING_SYSTEM=1.
- Bit 0 of machine data:

  MD20110 $MC_RESET_MODE_MASK

  set in every channel
- All channels in the reset state
- Axes are not being traversed via JOG, DRF or the PLC
- Constant grinding wheel peripheral speed (GWPS) is not active

are met on all connected NCUs, the switchover is valid for all NCUs. If one of the switchover conditions for the system of units is not fulfilled on one or more NCUs, none of the NCUs will switch to the other unit system.

Actions such as part program start or mode change are disabled for the duration of the switchover.

Different systems of units

It is only possible to use different unit systems in time intervals where there are no cross-NCU interpolations even though the link grouping is installed. The required settings as described in

References: /FB1/ Function Manual, Basic Functions; Velocities, Setpoint/Actual Value Systems, Control Loop Control (G2)

must be performed NCU-specifically by means of G codes.

2.14 Supplementary conditions

2.14.1 Several operator panels and NCUs with control unit management option

Configuration

The number of configurable control units/NCUs is only limited by the availability of bus addresses on the individual bus segments of the different bus types.

The control unit management option permits linking of up to **9 control units** on up to **9 NCUs**.

2.14.2 Several operator panel fronts and NCUs, standard functionality

Possible configurations

- Configuration "2 operator panels and 1 NCU"
  One of the operator panels must be an OP030.
- Configuration "1 operator panel and up to 4 NCUs"
  1 additional operator panel per NCU can be operated locally. If the link is set up via MPI, only NCUs whose NC address can be specified can be used.

Operation of the M:N link

via the channel menu (see Section "Operator interface") selected via the "Channel switchover" key.

The precondition for the channel menu is configuration via the NETNAMES.INI file (see /IAD/, Commissioning Manual 840D).

The channel menu function is an option.

Bus connection

Address space: 0, ..., 31

Note

If an address > 15 is used, all components connected to the bus must be capable of processing addresses between 0 and 31.

2.14.3 Link axes

Availability

1. Precondition is that the NCUs are networked with link modules.
2. The **link axis** function is an option which is necessary for each link axis (max. 32).
3. The **axis container** function is an option which is necessary for each container.

References: /PHD/ 840D NCU Configuration Manual

⚠ WARNING
In the context of Safety Integrated (SI) the boundary conditions must be observed, see brief description, Chapter NCU-Link.

2.14.4 Axis container

Availability

Axis container is an option. In cases where axis containers are configured for link axes, the supplementary conditions for these containers defined in Section "Link axes" also apply.

⚠ WARNING
In the context of Safety Integrated (SI) the boundary conditions must be observed, see brief description, Chapter NCU-Link.

2.14.5 Lead link axis

Availability

Cross-NCU coupling with lead link axis is based on an NCU link. Therefore, the NCU link options must be installed.

⚠ WARNING
In the context of Safety Integrated (SI) the boundary conditions must be observed, see brief description, Chapter NCU-Link.

2.14.6 NCU link with different interpolation cycles

Availability

NCU link with different interpolation cycles is an option.

All requirements that apply for link axes must be fulfilled.

If the non-local axes - seen from the point of view of the NCU with the fast interpolation cycle - are container axes (e.g. with multi-spindle turning machines) the corresponding option must also be installed.

2.15 Examples

2.15.1 Configuration file NETNAMES.INI with control unit management option

A sample configuration file NETNAMES.INI for the MMC 1 control unit for a system with four NCUs on the OPI is outlined below.

See Section "Structure of configuration file" for details.

Note

The marginal notes (bold print) on the left of the page serve to structure the information and are not part of the file.

; NETNAMES.INI Example 1 Start

HMI identification

; Identification entry

[own]

owner = MMC_1

PCU-NCU connections

; Connection entry

[conn MMC_1]

conn_1 = NCU_1 ; NCU 1

conn_2 = NCU_2 ; NCU 2

conn_3 = NCU_3 ; NCU 3

conn_4 = NCU_4 ; NCU 4

Bus identification

```
[param network]
bus = OPI; OPI bus (1.5 Mbaud)
```

HMI description

```
[param MMC_1]

mmc_typ         = 40          ; = 0100 0000: HMI is server and main control panel
mmc_bustyp      = OPI         ; Bus the HMI is attached to
mmc_address     = 10          ; HMI address
mstt_address    = 6           ; Address of the MCP to be switched simultaneously
name            = MMC_LINKS   ; Name of operator panel
start_mode      = ONLINE      ; HMI is connected to the DEFAULT NCU in online
                                mode
                              ; on power up in accordance with channel data, see
                                below
```

Description of NCU components

```
[param NCU_1]
type = NCU_572 ; NCU type
nck_address = 20 ; Address j of NCU component on the bus
plc_address = 20 ; Address p of PLC component on the bus
name = NCU1 ; name of NCU
[param NCU_2]
type = NCU_572 ; NCU type
nck_address = 21 ; Address j of NCU component on the bus
plc_address = 21 ; Address p of PLC component on the bus
name = NCU2 ; name of NCU
[param NCU_3]
type = NCU_572 ; NCU type
nck_address = 22 ; Address j of NCU component on the bus
plc_address = 22 ; Address p of PLC component on the bus
name = NCU3 ; name of NCU
[param NCU_4]
type = NCU_572 ; NCU type
nck_address = 23 ; Address j of NCU component on the bus
plc_address = 23 ; Address p of PLC component on the bus
name = NCU4 ; name of NCU
; End of descriptive entry
```

Channel data

Sample of a channel menu configuration with M:N assignment option:

[chan MMC_1]

```
DEFAULT_logChanSet = G_1      ; Group to be set on power up
DEFAULT_logChan = K_1_1       ; Channel to be set on power up
ShowChanMenu = TRUE           ; Display channel menu
```

List of channel groups:

```
logChanSetList    = G_1, G_2, G_3, G_4
[G_1]
logChanList       = K_1_1, K_1_2      ; Group G_1 channels
[G_2]
logChanList       = K_2_1, K_2_2      ; Group G_2 channels
[G_3]
logChanList       = K_3_1, K_3_2      ; Group G_3 channels
[G_1]
logChanList       = K_4_1, K_4_2      ; Group G_4 channels
[K_1_1]
logNCName         = NCU_1             ; 1st channel of 1st group
ChanNum           = 1
[K_1_2]
logNCName         = NCU_1             ; 2nd channel of 1st group
ChanNum           = 2
[K_2_1]
logNCName         = NCU_2             ; 1st channel of 2nd group
ChanNum           = 1
[K_2_2]
logNCName         = NCU_2             ; 2nd channel of 2nd group
ChanNum           = 2
[K_3_1]
logNCName         = NCU_3             ; 1st channel of 3rd group
ChanNum           = 1
[K_3_2]
logNCName         = NCU_3             ; 2nd channel of 3rd group
ChanNum           = 2
[K_4_1]
logNCName         = NCU_4             ; 1st channel of 4th group
ChanNum           = 1
[K_4_2]
logNCName         = NCU_4             ; 2nd channel of 4th group
ChanNum           = 2
```

End of NETNAMES.INI example 1.

2.15.2 User-specific reconfiguring of PLC program control unit switchover

Introduction

The solution outlined roughly below should be selected only if at least one of the following configuring requirements is applicable:

- Suppression strategy which differs from standard functionality
- Operating mode switchover which differs from standard functionality
- Independent handling of override switch for switchover of control unit
- Existence of a 2nd machine control panel on an MMC

Method of description:

1. Description of operational sequences
2. Description of available functionality (Defines)
3. Graphical representation of sequences in diagram form

Implementation details can also be obtained from the standard configuration which is included in the toolbox.

2.15.2.1 Description of operational sequences (overview)

Overview

PCU sends request

A PCU would like to link up with an NCU and sends this request to the PLC of the relevant NCU.

PCU coming

A PCU goes online to an NCU, i.e. it links up to the NCU.

PCU going

A PCU aborts the link to an NCU.

Suppression

A PCU must abort the link with an NCU because another operator panel wants to go online to the same NCU.

Operating focus switchover in server mode

A server maintains a permanent link to the NCUs to which it is assigned. The operator can switch the operating focus from one NCU to another without interrupting the existing link.

Active/passive operating mode

An online operator panel can operate in two different modes:

Active mode: Operator can control and monitor

Passive mode: Operators sees header information and the "passive" identifier.

MCP switchover

As an option, an MCP assigned to the operator panel can be switched over at the same time.

2.15.2.2 Description of operational sequences (details)

Introduction

The operational sequences are described using identifiers for defined, logical functions (example: `OFFL_REQ_OP/ OK`) whose programming application has been described earlier in this section. The functions are coded in accordance with Section "Defined logical functions/defines". The functions store values in the interface which can be addressed from the PLC and the HMI. An operator panel utilizes the online-request interface while competing for the use of an online interface. Operator panels which are already linked to an NCU utilize one of the two available online interfaces. Details of these interfaces can be found in Section "Signal descriptions" and in

References: Lists

In order to illustrate complete operating sequences, the description covers HMI activities which cannot be influenced as well as **modifiable PLC activities**.

Operator panel sends request

If the operator panel is already linked online to an NCU (online NCU) and would like to communicate with another NCU (target NCU), it must first notify the PLC of the online NCU that it wishes to switch over to the target NCU.

It sends the offline request `OFFL_REQ_OP/ OK` to the online PLC.

`OFFL_CONF_OP/OK`:

Online PLC has received the offline request. HMI can now send a request to the target PLC.

OFFL_CONF_OP/PLC_LOCKED

Online PLC has received the offline request. The operator panel switchover is disabled in the HMI-PLC interface. The operator panel cannot link up with another NCU and must remain online.

On receipt of the positive acknowledgement OFFL_CONF_OP/OK, the operator panel sends its online request to the target PLC of the relevant NCU by transmitting its *client identification*.

Client identification: Unique HMI identifier comprising bus type and bus address. (ONL_REQUEST DB19, DBW100)

The target PLC sends the operator panel a positive or negative acknowledgement:

Pos. acknowledgement: Target PLC returns the client identification to the operator panel. (ONL_CONFIRM, DB19, DBW102). Operator panel occupies the online-request interface with its parameters (client identification, MMC type, MCP address). Operator panel can go online once it has received the online permission from the target PLC.

Neg. acknowledgement: Target PLC does not return the client identification to the operator panel. (ONL_CONFIRM, DB19, DBW102 not identical to client identification of requesting operator panel). Operator panel cannot go online.

Example:

Another operator panel is currently switching over to the same NCU. This switchover operation must not be interrupted. The operator panel remains online to the online NCU.

Once the operator panel has received a positive acknowledgement from the PLC, it may need to suppress another online operator panel. The operator panel then receives positive/negative online permission from the PLC.

Positive:

ONL_PERM/OK

On receipt of positive online permission (DB19, DBB 108, 109), the operator panel can go online. An HMI-PLC interface is allocated to the operator panel at the same time as online permission. (1 or 2, details can be found in the interface description in Section "Signal descriptions").

Negative:

ONL_PERM/MMC_LOCKED

The requesting operator panel cannot go online. Two operator panels on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot suppress either of the two operator panels.

ONL_PERM/PLC_LOCKED

The requesting operator panel cannot go online. The operator panel switchover is disabled in the HMI-PLC interface.

ONL_PERM/PRIO_H

The requesting operator panel cannot go online. Two operator panels that are both higher priority than the requesting operator panel are connected online to the NCU. The PLC cannot suppress either of the two operator panels.

Operator panel coming

Once the operator panel has sent an online request to the target PLC and received online permission from it, it can set up a link to the target NCU.

It goes online and notifies the PLC with (station active) S_ACT/CONNECT that it has linked up with the NCU.

The operator panel sets up its sign of life signal in accordance with the allocated interface.

The operator panel then requests

- in the case of operator panel front: active operating mode on the target NCU,
- in the case of server: operating focus on the target NCU.

The PLC then activates HMI sign-of-life monitoring for the new operator panel.

See: Active/passive operating mode

See: Operating focus switchover in server mode

Operator panel going

An operator panel aborts communication with an NCU.

Communication can be aborted for two different reasons:

1. The operator wishes to switch the operator panel to another NCU. The operator panel has sent an online request to the target PLC and received online permission (ONL_PERM/OK). It has notified the online PLC of its intention to switch over with OFFL_REQ_OP/OK and received a positive acknowledgement (OFFL_CONF_OP/OK). Due to the switchover to the target NCU, the HMI sign of life in the online PLC is changed from TRUE to FALSE. The falling edge combined with the sequence described above signals to the online PLC that the operator panel has broken off the link to the online NCU. If an MCP is assigned to the operator panel and activated, it is now deactivated by the PLC. Passive operating mode is set in the PLC for the operator panel which has been suppressed.

   See: Active/passive operating mode
2. The operator panel is suppressed from the PLC by the online request from another operator panel. See "Suppression".

Suppression

Two operator panels are linked online to an NCU, each is occupying an HMI-PLC interface. A third operator panel would like to go online.

The PLC must suppress one of the two operator panels according to a predefined strategy. It sends the operator panel to be suppressed an offline request (OFFL_REQ_OP/OK) to abort communication with the NCU. The operator panel returns a positive or negative acknowledgement to the PLC:

Positive:

OFFL_CONF_PLC/OK

Operator panel breaks off the link to the NCU and switches to the offline state.

The HMI sign of life in the PLC changes from TRUE to FALSE.

The falling edge combined with the sequence described above signals to the online PLC that the operator panel has broken off the link to the online NCU. If an MCP is assigned to the operator panel and activated, it must now be deactivated by the PLC.

The PLC also ceases to monitor the HMI sign of life signal.

Passive operating mode is set in the PLC for the operator panel which has been suppressed.

See "Active/passive operating mode" further below.

Negative: OFFL_CONF_PLC/MMC_LOCKED

Processes that cannot be interrupted are running on the operator panel (e.g. operation via RS-232 or data transfer between NCU and PCU).

The operator panel remains online to the current NCU.

Operating focus switchover in server mode

A server maintains a permanent link to the NCUs to which it is assigned. The operator can switch the operating focus from one NCU to another without interrupting the existing link.

If the operator wishes to switch the operating focus to another NCU, the focus PLC and target PLC must first be interrogated to determine whether they will permit a focus switchover.

The operator panel first sends the focus offline request signal (OFFL_REQ_FOC/OK) to the focus PLC.

After a positive acknowledgement (OFFL_CONF_FOC/OK) from the focus PLC, the operator panel sends query signal ONL_REQ_FOC/OK regarding focus changeover to the target PLC.

After the operator panel has received permission from the target PLC to switch the operating focus (ONL_PERM_FOC/OK), the operator panel logs off from the focus PLC with S_ACT/DISC_FOCUS and switches the focus to the target PLC.

The operator panel must finally request active operating mode in the target NCU. The previous focus PLC must set active operating mode for this HMI-PLC interface after receiving S_ACT/DISC_FOCUS and deactivate any active MCP assigned to the operating panel which has gone offline.

See: Active/passive operating mode

Active/passive operating mode

After an operator panel has gone online to an NCU, it can assume one of two different operating states:

Active mode: Operator can control and monitor

Passive mode: Operator sees header information and the "passive" status identifier.

After switching to an NCU, it first requests active operating mode in the online PLC.

If two operator panels are linked online simultaneously to an NCU, one of the two is always in active mode and the other in passive mode.

The operator can request active mode on the passive operator panel at the press of a button.

If an MCP has been configured for the online operator panels, the MCP of the active operator panel is switched on. The MCP of the passive operator panel is deactivated, i.e. only one MCP is active at a time on an NCU.

Four signals are provided in the HMI-PLC interface for each of the two online operator panels. These signals are used by the PLC to control operating mode changeovers.

Table 2- 6 Signals (x = 1, 2: 1st or 2nd HMI-PLC interface)

HMI-PLC interface	Value	Meaning
MMCx_ACTIVE_REQ	FALSE → TRUE	HMI to PLC: Passive operator panel requests active operating mode
	TRUE → FALSE	PLC to HMI: Request received
MMCx_ACTIVE_PERM	FALSE → TRUE	PLC to HMI: Passive operator panel can change to active operating mode
	TRUE → FALSE	PLC to HMI: Active operator panel must change to passive operating mode
MMCx_ACTIVE_CHANGED	FALSE → TRUE	HMI to PLC: Operator panel has completed changeover from passive to active mode
	TRUE → FALSE	HMI to PLC: Operator panel has completed changeover from active to passive mode
MMCx_CHANGE_DENIED	FALSE → TRUE	HMI to PLC or PLC to HMI depending on interface: Operating mode cannot be changed due to uninterruptible processes on active operator panel.
	TRUE → FALSE	HMI to PLC or PLC to HMI depending on interface: Acknowledgement of MMCx_CHANGE_DENIED(FALSE->TRUE)

An example of how operating modes can be switched over is described in the following sequence.

Two operator panels online to one NCU, MMC_1 in active operating mode, MMC_2 in passive operating mode, operator requests active operating mode on MMC_2.

Signal state for this case:

MMC_1	VALUE	MMC_2	Value
MMC1_ACTIVE_REQ	FALSE	MMC2_ACTIVE_REQ	FALSE
MMC1_ACTIVE_PERM	TRUE	MMC2_ACTIVE_PERM	FALSE
MMC1_ACTIVE_CHANGED	TRUE	MMC2_ACTIVE_CHANGED	FALSE
MMC1_CHANGE_DENIED	FALSE	MMC2_CHANGE_DENIED	FALSE

MMC_2 requests active operating mode and sets MMC_2_AKTIVE_REQ = TRUE.

The PLC acknowledges the request from MMC_2 with MMC_2_ACTIVE_REQ = FALSE.

The PLC then requests MMC_1 to change to passive operating mode with MMC1_ACTIVE_PERM = FALSE.

We must differentiate between two cases here:

1. MMC_1 can change to passive operating mode:

   MMC_1 changes from active to passive operating mode and acknowledges the changeover with

   MMC1_ACTIVE_CHANGED = FALSE.

   If an MCP is assigned to the MMC and activated, it is now deactivated by the PLC.

   The PLC gives permission for a changeover to the active operating mode with MMC2_ACTIVE_PERM = TRUE.

   MMC_2 changes state and acknowledges with MMC2_ACTIVE_CHANGED = TRUE. If an MCP is assigned to MMC_2, it is now activated by the PLC.

2. MMC_1 cannot change to the passive operating mode (processes which do not permit a changover are running on MMC_1):

   MMC_1 sets MMC1_CHANGE_DENIED = TRUE, the change of state cannot be completed.

   The PLC acknowledges with MMC1_CHANGE_DENIED = FALSE and gives permission to MMC_1 to remain in active mode with MMC1_ACTIVE_PERM = TRUE. By sending MMC2_CHANGE_DENIED = TRUE, it notifies MMC_2 that MMC_1 cannot switch over to passive mode.

   MMC_2 then acknowledges with MMC2_CHANGE_DENIED = FALSE and remains in passive operating mode.

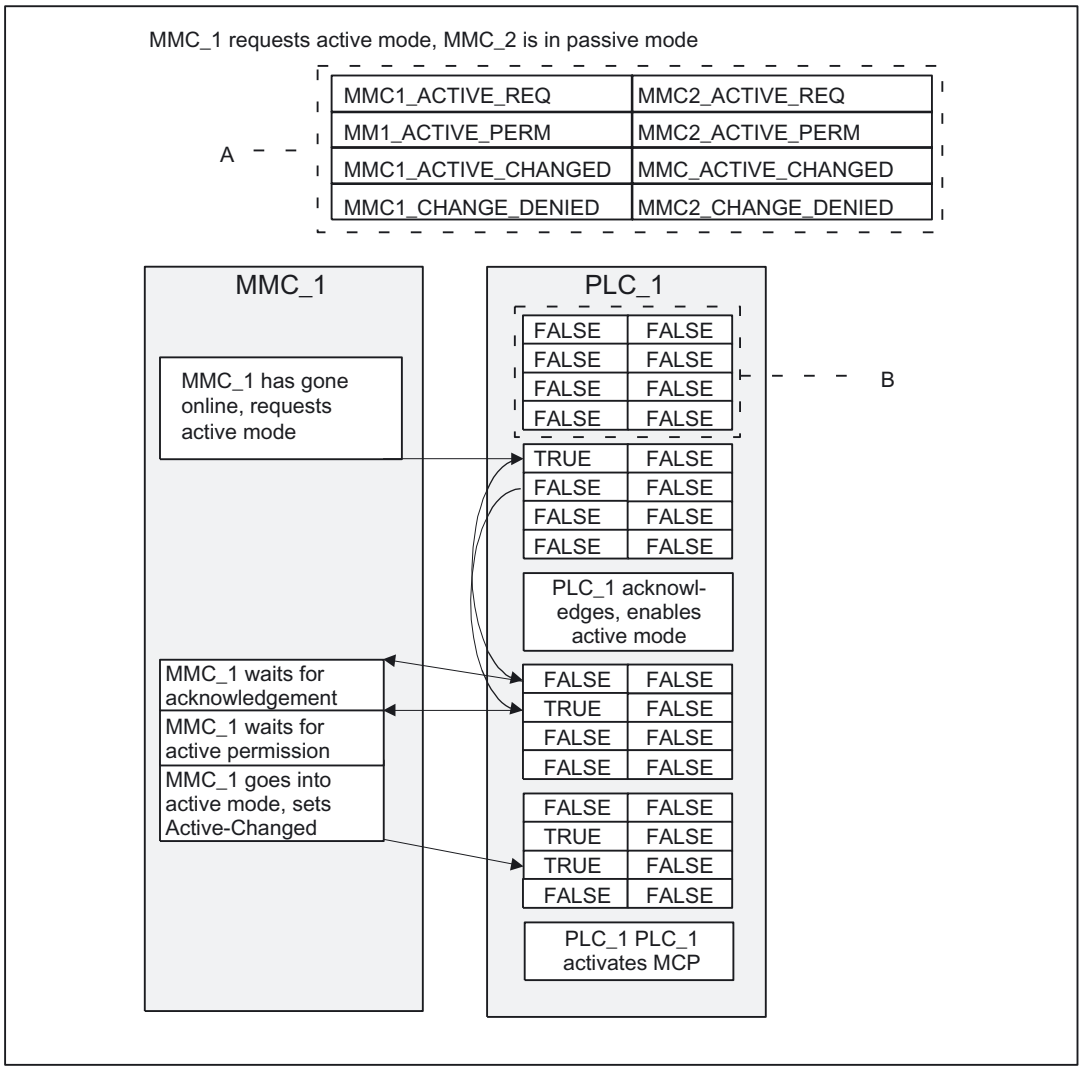


Figure 2-33 MMC_1 requests active mode, MMC_2 is in passive mode

Note for the reader

The arrangement of the signals of a block in box PLC_x (marked as B) corresponds to the arrangement of signal names in the header section (marked as A). Blocks B repeat in box PLC_x from top to bottom as a function of time.

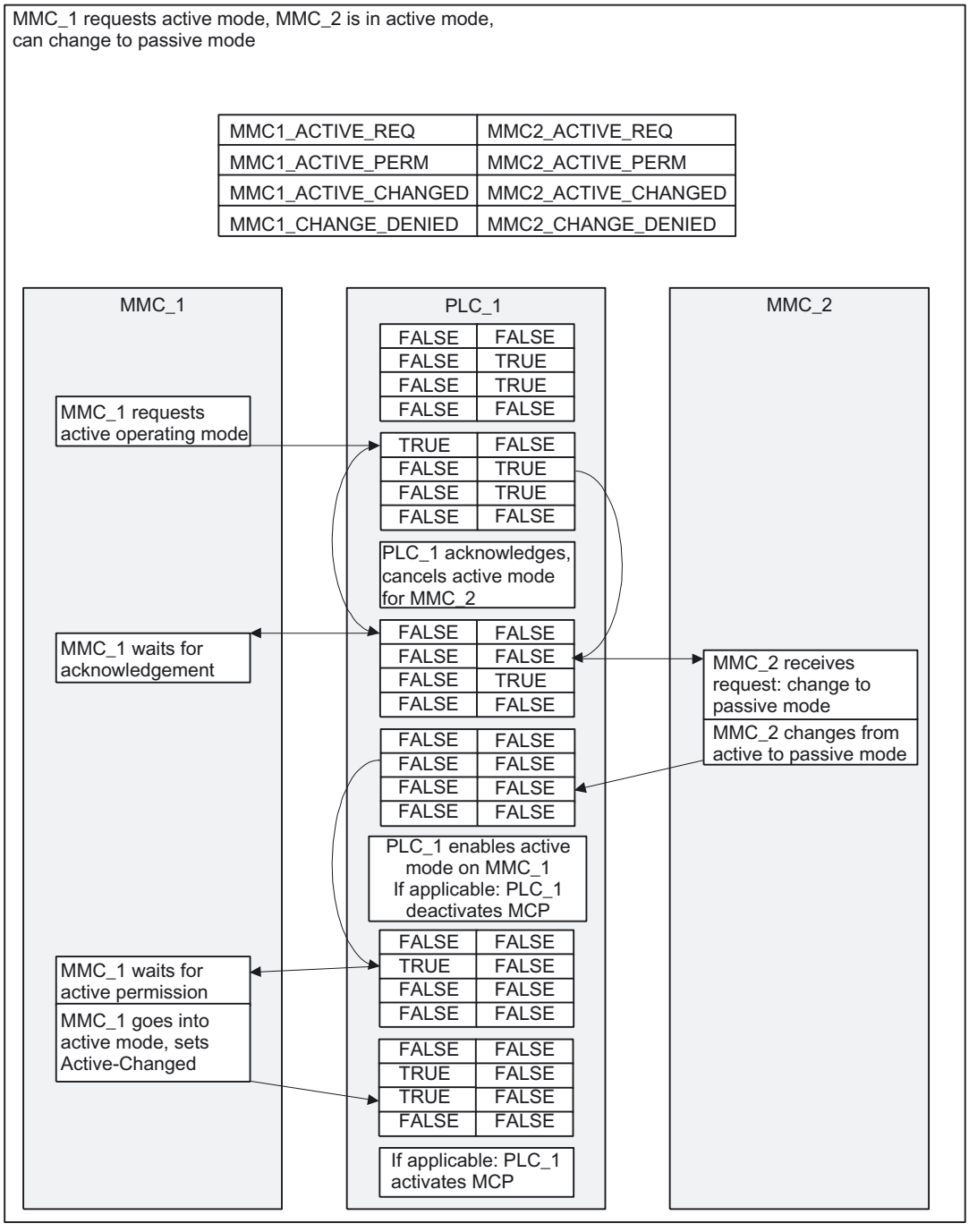


Figure 2-34 MMC_1 requests active mode, MMC_2 is in active mode, can change to passive mode

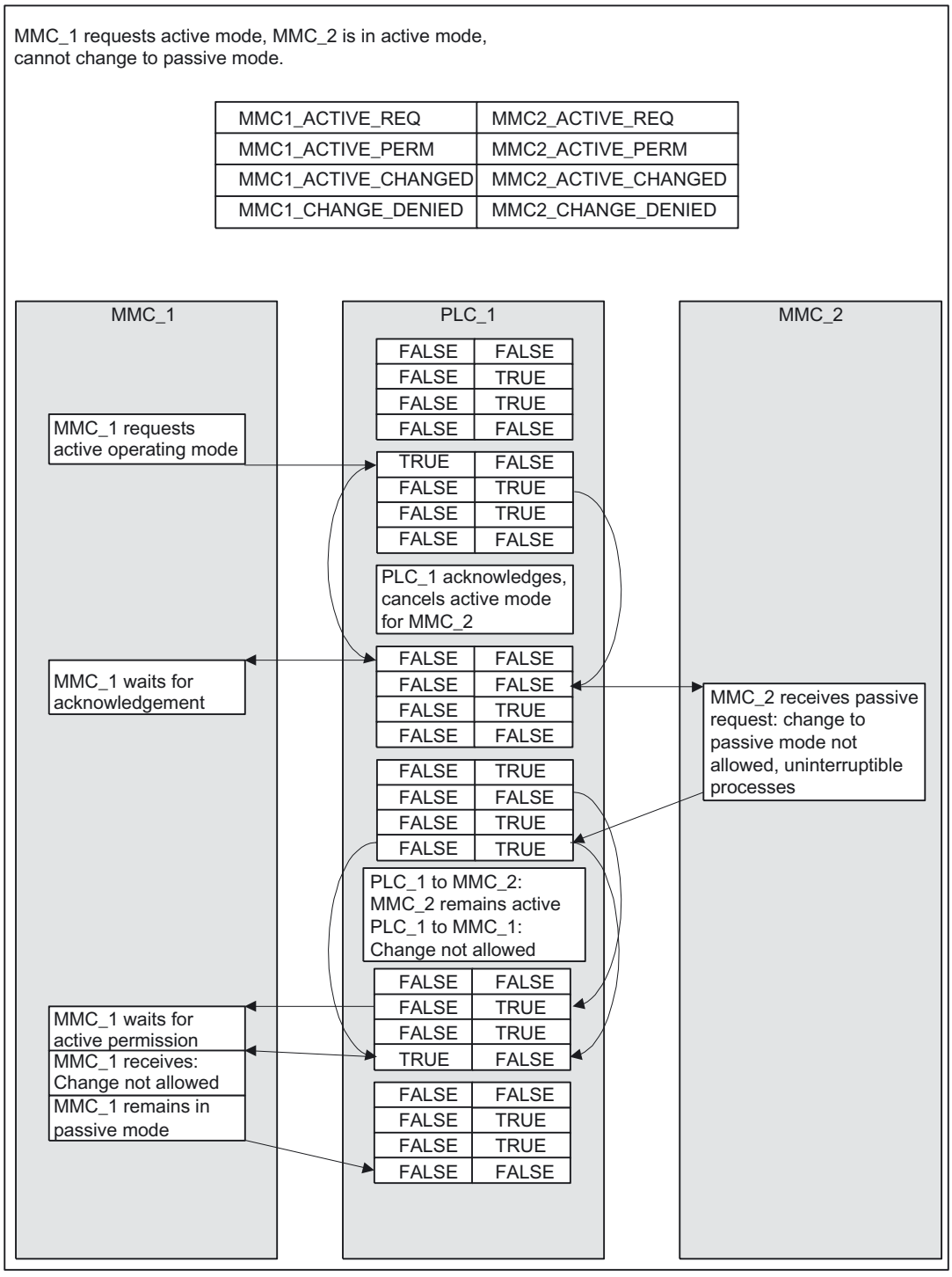


Figure 2-35 MMC_1 requests active mode, MMC_2 is in active mode, but cannot change to passive mode

MCP SWITCHOVER

A control unit consists of an operator panel and an MCP; these can both be switched over as a unit.

If an MCP has been configured for the operator panel in configuring file NETNAMES.INI, it will be activated and deactivated with the operator panel.

The operator panel of whichever MMC is currently in active operating mode is activated.

In other words, only **one** MCP is ever active at any time on an NCU.

The MCP is activated by the PLC:

- Operator panel changes to active operating mode. (signal MMCx_ACTIVE_CHANGED: FALSE -> TRUE, x = 1,2 first or second HMI-PLC interface)

The MCP is deactivated by the PLC

- Operator panel changes to passive operating mode.

  (signal MMCx_ACTIVE_CHANGED: TRUE -> FALSE, x = 1,2 first or second HMI-PLC interface)

- Operator panel goes offline by means of switchover or suppression

  The HMI sign of life changes from TRUE to FALSE when an operator panel goes offline. After the edge change, the PLC deactivates the allocated MCP.

- Server HMI disconnects operating focus from the current NCU and switches it over to another. The server transmits S_ACT/ DIS_FOCUS as the last signal on its own HMI-PLC interface. The PLC then deactivates the corresponding MCP.

2.15.2.3 Defined logical functions/defines

Note

Please refer to Section "Defined logical functions/defines" for the legal values for bus type, functions/status and additional information, plus permissible combinations of status and additional information. The logical identifiers of functions are used in the following diagrams.

2.15.2.4 Graphical representation of function sequences

Overview

The diagrams below describe how an operator station is switched over (switchover from NCU_1 to NCU_2).

The first five diagrams describe the switchover operation for an operator station and the next three the switchover operation for a server.

If an operator panel (MMC) in the offline state wants to go online on an NCU (e.g. on power up), the sequence OFFL_REQ_OP (...), or OFFL_CONF_OP(...) is not necessary.

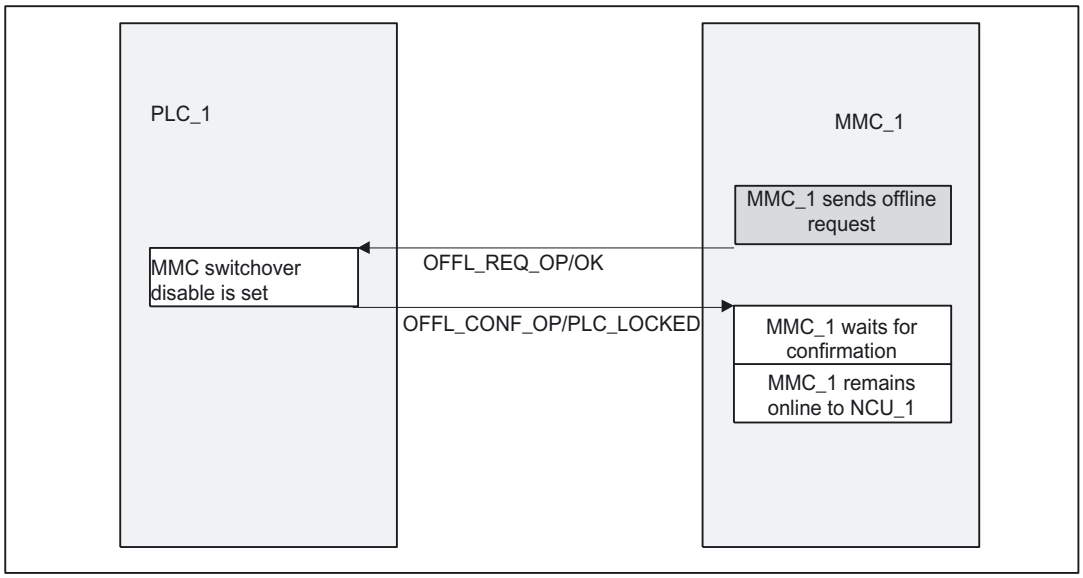


Figure 2-36 MMC_1 is linked online to NCU_1 and wants to switch over to NCU_2, switchover disable is set in PLC_1

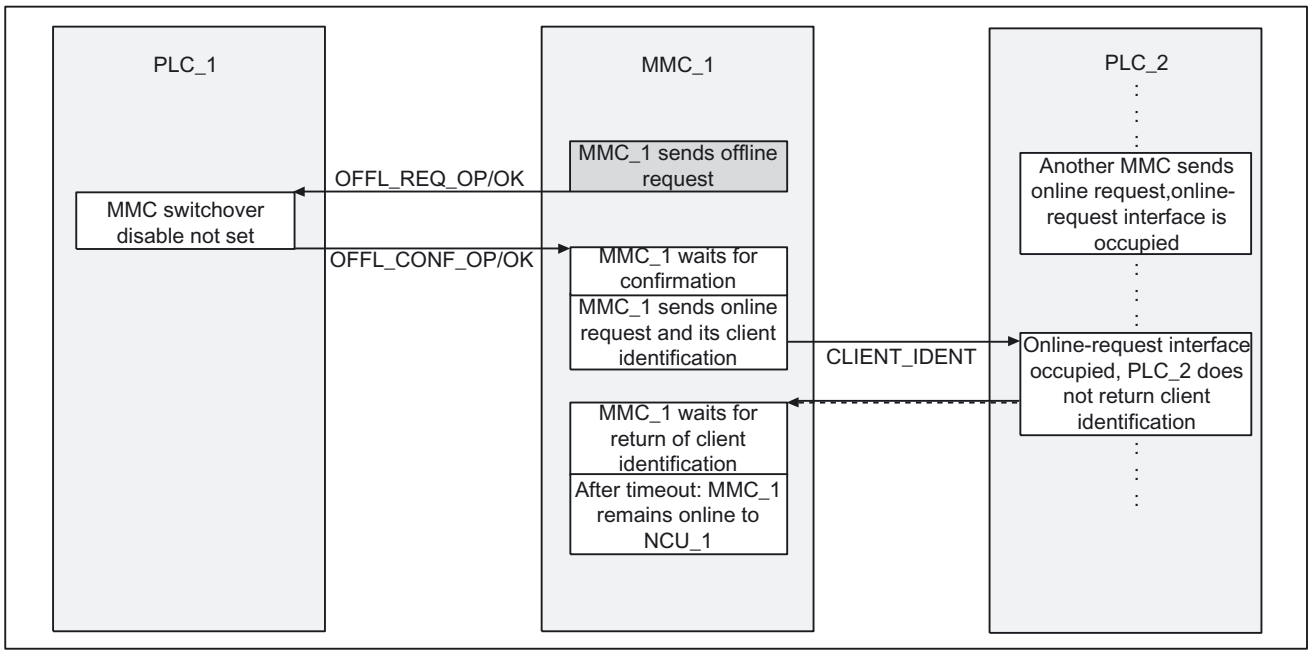


Figure 2-37 MMC_1 online to NCU_1, MMC_1 wants to switch over to NCU_2, online-request interface in PLC_2 occupied by another MMC

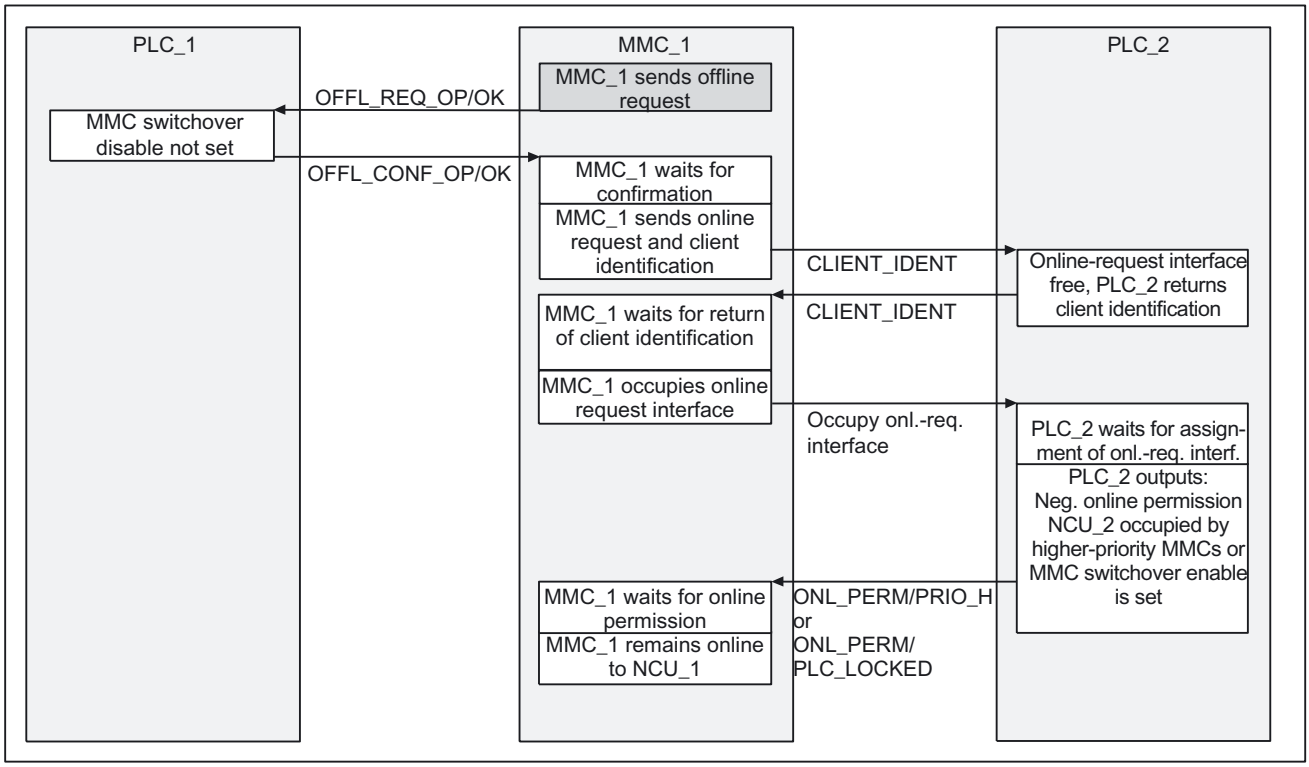


Figure 2-38 MMC_1 online to NCU_1, MMC_1 wants to switch over to NCU_2, but does not receive permission from PLC_2

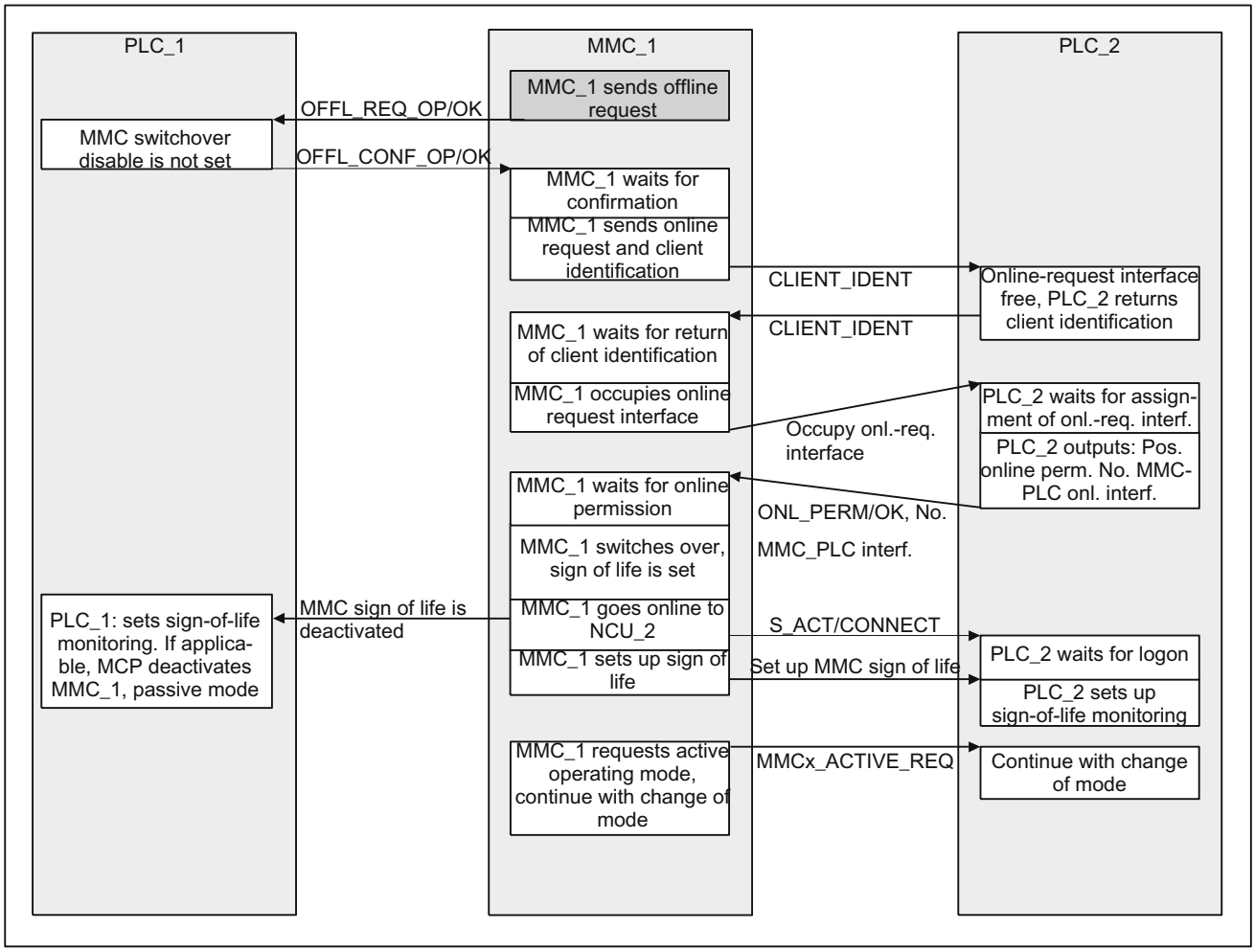


Figure 2-39 MMC_1 online to NCU_1, MMC_1 switches over to NCU_2 (no suppression)

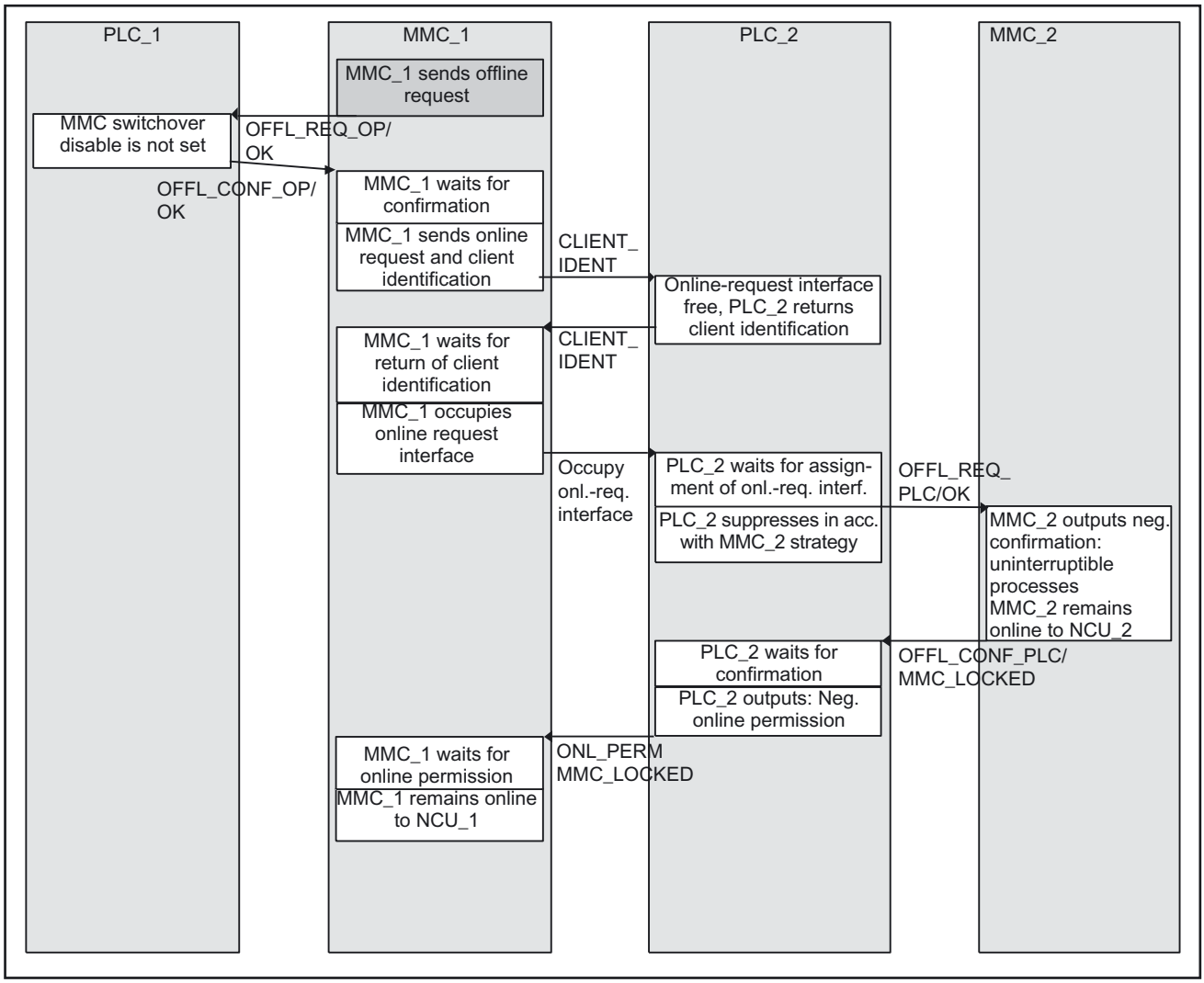


Figure 2-40 MMC_1 online to NCU_1, MMC_2 online to NCU_2, MMC_1 wants to switch over to NCU_2, but MMCs executing uninterruptible processes are online to NCU_2

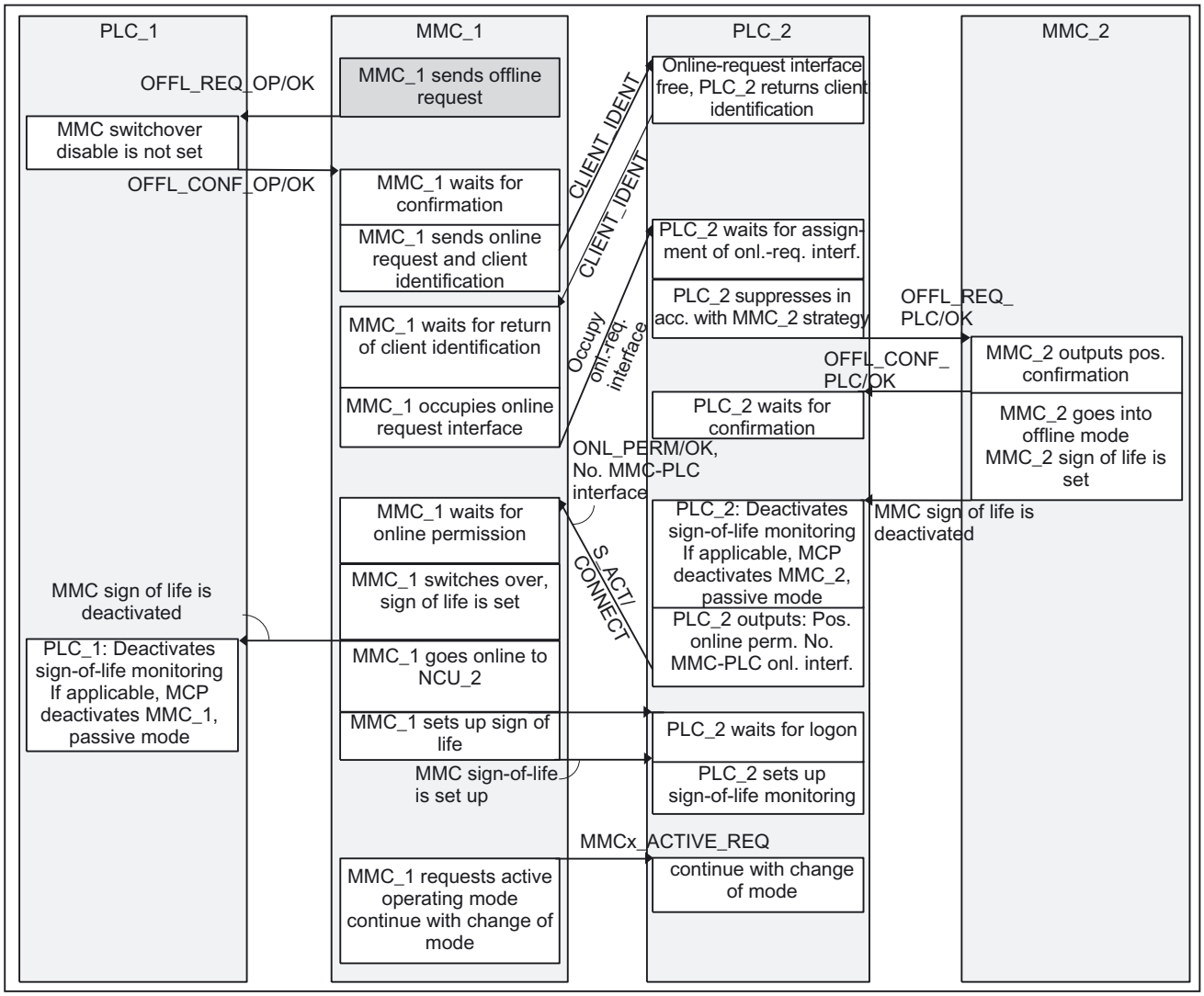


Figure 2-41 MMC_1 online to NCU_1, MMC_2 online to NCU_2, MMC_1 switches from NCU_1 to NCU_2, MMC_2 is suppressed

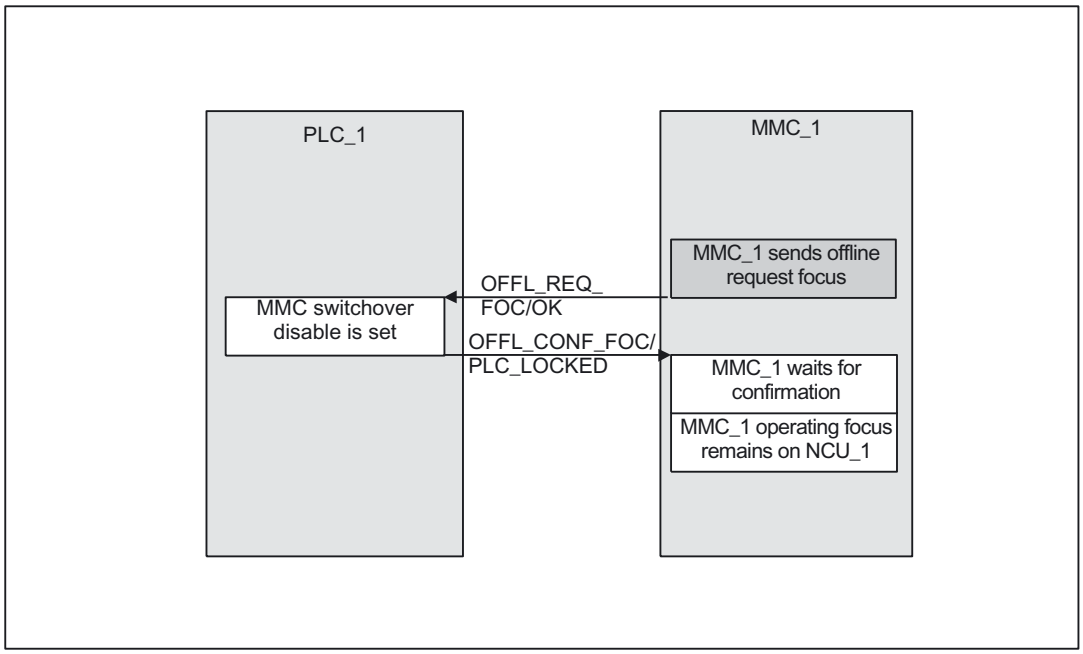


Figure 2-42 MMC_1 server, wishes to switch operating focus from NCU_1 to NCU_2, switchover disabled in PLC_1

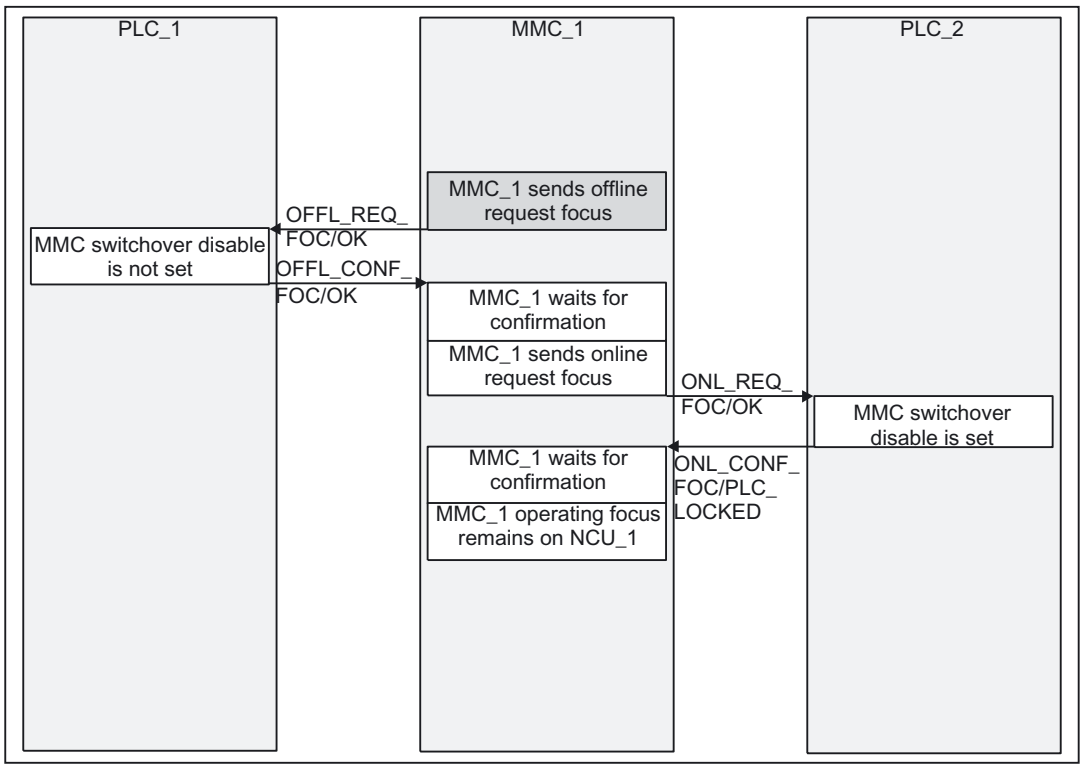


Figure 2-43 MMC_1 is server, wishes to switch operating focus from NCU_1 over to NCU_2, switchover is disabled in PLC_2

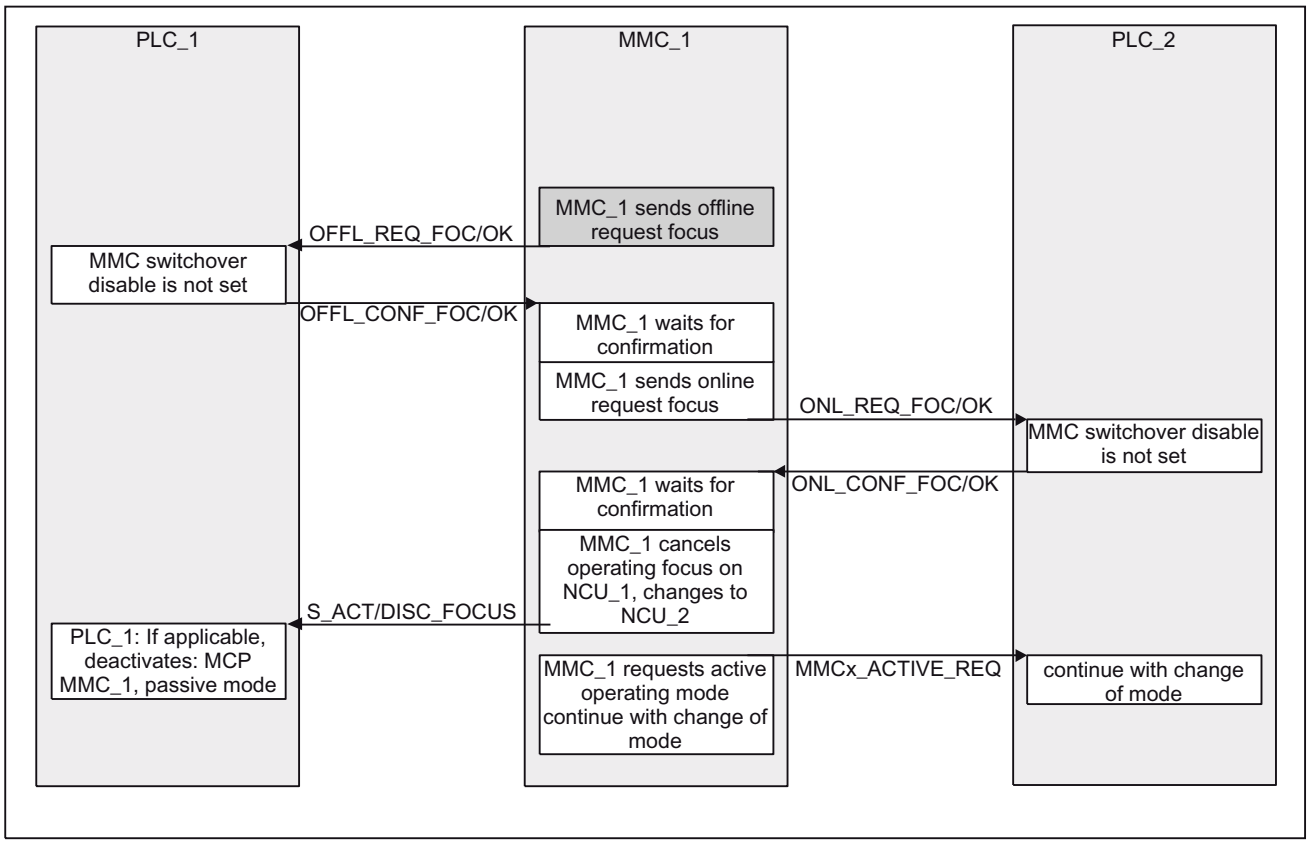


Figure 2-44 MMC_1 is server, wishes to switch operating focus from NCU_1 over to NCU_2, switchover not disabled in PLCs, MMC_1 can switch operating focus

2.15.3 Configuration file NETNAMES.INI, standard functionality

2.15.3.1 Two operator panel fronts and one NCU

A sample configuration file for the second control unit is given below for a system consisting of two control units and one NCU on the OPI.

```
; NETNAMES.INI Example 2 Start

; Identification entry
[own]
owner = MMC_2
; Connection entry
[conn MMC_1]
conn_1 NCU_1

[conn MMC_2]
conn_1 NCU_1
; Descriptive entry
[param network]
bus = opi

[param MMC_1]
mmc_address = 1

[param MMC_2]
mmc_address = 3

[param NCU_1]
nck_address = 13
plc_address = 13

; NETNAMES.INI example 2 End
```

2.15.3.2 One operator panel front and three NCUs

A sample configuration file is given below for a system consisting of one control unit and three NCUs on the OPI.

Any adaptations which may need to be made are described in Section "Configurations".

```
; NETNAMES.INI Example 3 Start
; Identification entry:
[own]
owner = MMC_1

; Connection entry: For the planned number of up to 3 connections
[conn MMC_1]
conn_1= NCU_1
conn_2= NCU_2
conn_3= NCU_3

; Descriptive entry: The name of the network is clearly stated
[param network]
bus= opi

[param MMC_1]
name= any_name
type= MMC_100
mmc_address= 1

[param NCU_1]
name= any_name1
type= ncu_572
nck_address= 12
plc_address = 12

[param NCU_2]
name= any_name2
type= ncu_573
nck_address= 14
plc_address= 14

[param NCU_3]
name= any_name3
```

type= ncu_573

nck_address= 15

plc_address= 15

; NETNAMES.INI, example 3 End

2.15.4 Quick M:N commissioning based on examples

Introduction

The MPI/OPI bus network rules are not described.

See **References**:

/BH/, Operator Components Manual

Three examples are used to demonstrate the steps involved in starting up an M:N grouping. Each description begins by presenting a configuration.

2.15.4.1 Example 1

Hardware configuration

The hardware comprises the following components:

- 1 operator panel (PCU50 with HMI Advanced, operator panel, machine control panel)
- Two NCUs with two channels each

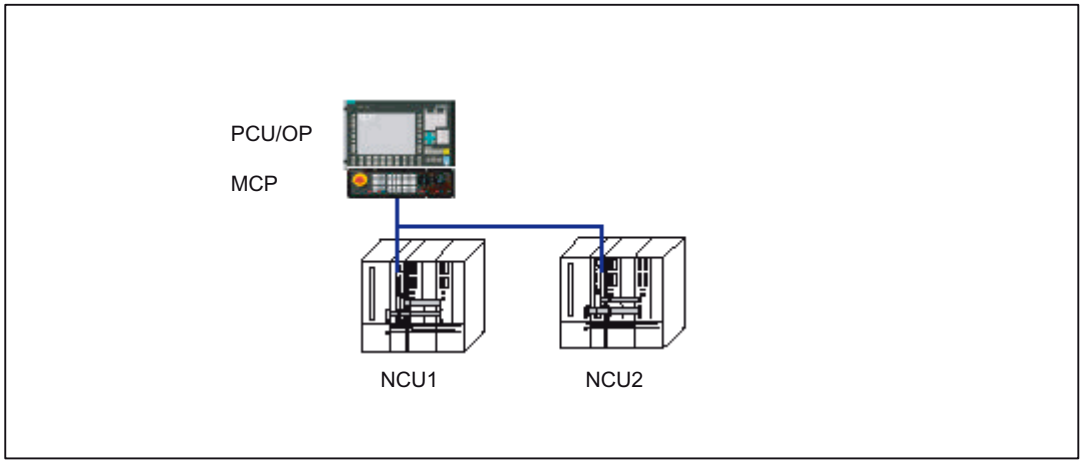


Figure 2-45 One operator panel for two NCUs

Step 1:

Configuration file NETNAMES.INI

The following entries are made in this example:

```
[own]
owner = MMC_1
; Connection entry
[conn MMC_1]
conn_1 = NCU_1
conn_2 = NCU_2
; Extcall not required for a PCU
; Network parameters
[param network]
bus= opi
; HMI descriptions
[param MMC_1]
mmc_address = 1
; All other parameters not required
; NCU components descriptive entry
[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1

[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2

; Channel data
[chan MMC_1]
DEFAULT_logChanSet = Station_1
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_2
```

```
[Station_1]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_1
ChanNum = 1
[N1_K2]
logNCName = NCU_1
ChanNum = 2

[Station_2]
logChanList = N2_K1, N2_K2
[N2_K1]
logNCName = NCU_2
ChanNum = 1
[N2_K2]
logNCName = NCU_2
ChanNum = 2
; End
```

Step 2:

Load file NETNAMES.INI

HMI Advanced/PCU50: Once the NETNAMES.INI file has been created, it is transferred into the USER directory of the PCU

Step 3:

Set the NCK bus addresses

1. Enter the following in the "Operator panel front interface parameters" input screen via "Commissioning → HMI → Operator panel":
   Connection: M:N (Select M:N instead of 1:1)
   NCK address: 22
   PLC address: 22
   according to NETNAMES.INI for NCU2 address 23
2. "Save"
3. Restart the PCU

Step 4:

PLC

An FB9 call is not required for this configuration, because no suppression or active/passive switching takes place.

Softkey label

The texts are transferred from the NETNAMES.INI file. No extra texts over and above those in NETNAMES.INI are required for the present example.

2.15.4.2 Example 2

Hardware configuration

The hardware comprises the following components:

- Operator panel 1 (PCU50 with HMI Advanced, operator panel, machine control panel)
- Operator panel 2 (PCU50 with HMI Advanced, operator panel, machine control panel)
- Two NCUs with two channels each

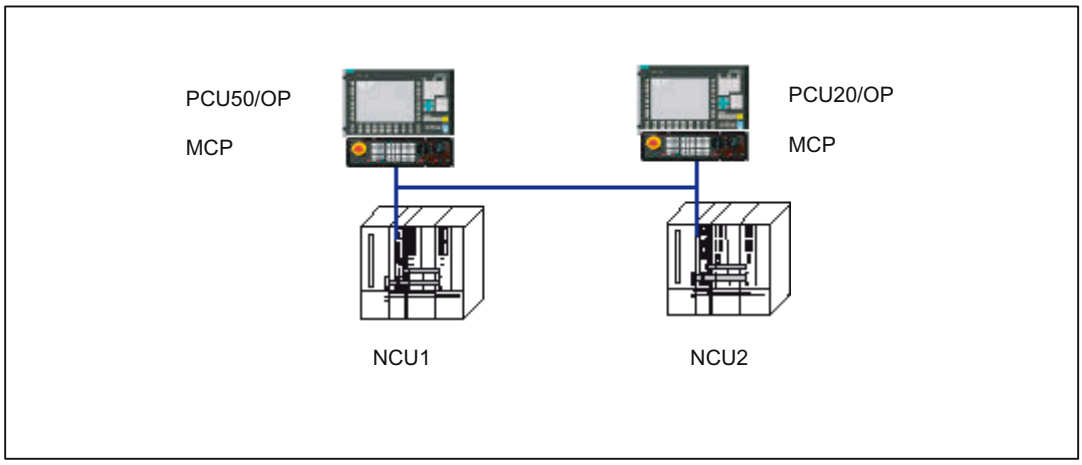


Figure 2-46 Two operator panels for two NCUs

Use

Operator panel 1 (server) and operator panel 2 can access NCU1 and NCU2.

Step 1a):

NETNAMES.INI configuration files

In this example, separate entries are input for the operator panels in NETNAMES.INI files.

Operator panel 1

Entries for HMI Advanced/PCU50:

```
[own]
owner = MMC_1
; Connection entry
[conn MMC_1]
conn_1 = NCU_1
conn_2 = NCU_2
EXTCALL_conns = conn_1, conn_2
; Network parameters
[param network]
bus= opi
; HMI descriptions
[param MMC_1]
mmc_type = 0x40
mmc_bustyp = OPI
mmc_address = 1
mstt_address = 6
name = MMC_Serv
start_mode = ONLINE
; NCU components descriptive entry
[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1

[param NCU_2]
type = NCU_573
nck_address = 23
plc_address = 23
name = NCU2
```

```
; Channel data
[chan MMC_1]
DEFAULT_logChanSet = Station_1
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_2

[Station_1]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_1
ChanNum = 1
[N1_K2]
logNCName = NCU_1
ChanNum = 2

[Station_2]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_2
ChanNum = 1
[N1_K2]
logNCName = NCU_2
ChanNum = 2
; End
```

Step 2a):

Load file NETNAMES.INI

HMI Advanced/PCU50: Once the NETNAMES.INI file has been created, it is transferred into the USER directory of the PCU

Step 1b):

Operator panel 2

Entries for HMI Embedded/PCU20:

```
[own]
owner= PCU20
; Connection entry
[conn PCU20]
conn_1 = NCU_1
conn_2 = NCU_2
; Network parameters
[param network]
bus= opi
; HMI descriptions
[param PCU20]
mmc_typ = 0x10
mmc_bustyp = OPI
mmc_address = 2
mstt_address = 7
name = MMC_Neben
start_mode = OFFLINE
; NCU components descriptive entry
[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1
 [param NCU_2]
type = NCU_573
nck_address = 23
```

```
plc_address = 23
name = NCU2
; Channel data
[chan PCU20]
DEFAULT_logChanSet = Station_2
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_1
 [Station_1]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_1
ChanNum = 1
[N1_K2]
logNCName = NCU_1
ChanNum = 2
 [Station_2]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_2
ChanNum = 1
[N1_K2]
logNCName = NCU_2
ChanNum = 2
; End
```

Softkey label

In order to distinguish which NCU is to be addressed, texts for labeling the OP softkeys must be defined in file **chan.txt**:

```
//*Max. length of text 2*9 characters*/
//* Create new line with %n at the end of the first line*/
//* Name of channel area 1 and names of the channels of this area */
T_CHAN_AREA_1 "Stat_1"
T_CHAN_AREA_1_CHANNEL_1 "N1_K1"
T_CHAN_AREA_1_CHANNEL_2 "N1_K2"
//* Name of channel area 2 and names of the channels of this area */
T_CHAN_AREA_2 "Stat_2"
T_CHAN_AREA_2_CHANNEL_1 "N2_K1"
T_CHAN_AREA_2_CHANNEL_2 "N2_K2"
```

Step 2b:

PCU20

After the NETNAMES.INI and chan.txt files have been created, they are included in the *.abb file with the application.

Step 3:

Set the NCK bus addresses

HMI Advanced/PCU50:

1. Enter the following in the "Operator panel front interface parameters" input screen via "Commissioning → HMI → Operator panel":
   Connection: M:N (with key Select instead of 1:1)
   NCK address: 22
   PLC address: 22
   according to NETNAMES.INI for NCU2 address 23
2. "Save"
3. Restart the PCU

HMI Embedded/PCU20:

Transfer *.abb onto the system using a PC card and perform a software update.

Note

If you have forgotten to include the "chan.txt" file in *.abb, no labeled softkeys will be visible when you select the channel menu key. The selection function is available, however.

Step 4:

PLC

Include FB9 in the PLC user program. You will find more details after the examples below.

2.15.4.3 Example 3

Hardware configuration

The hardware comprises the following components:

- 1 operator panel (PCU50 with HMI Advanced, operator panel)
- 1 HT6
- 2 NCUs with two channels each

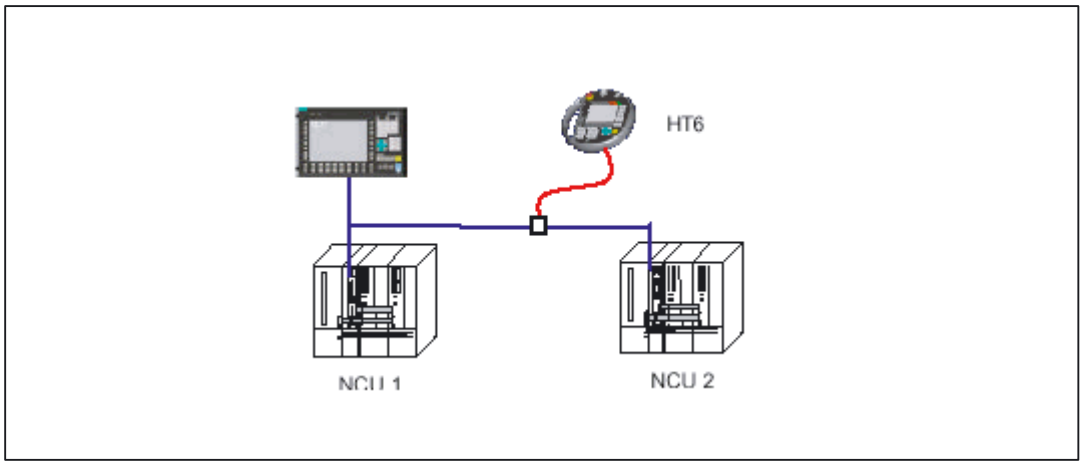


Figure 2-47 Operator panel and HT6 for two NCUs

Use

The operator panel (server) can access NCU1 and NCU2.

HT6 can only access NCU2.

Step 1a:

Create the NETNAMES.INI file for HMI Advanced/PCU 50

```
[own]
owner = MMC_1
; Connection part
[conn MMC_1]
conn_1= NCU_1
conn_2= NCU_2
EXTCALL_conns = conn_1, conn_2
; Network parameters
[param network]
bus= opi
```

```
; HMI descriptions
[param MMC_1]
mmc_type = 0x40
mmc_bustyp = OPI
mmc_address = 1
mstt_address = 255 ; 255 is necessary if no MCP
; is configured.
name = MMC_Serv
start_mode = ONLINE
; Description of NCU components
[param NCU_1]
type = NCU_573
nck_address = 22
plc_address = 22
name = NCU1
 [param NCU_2]
type =NCU_573
nck_address = 23
plc_address = 23
name = NCU2
; Channel data
[chan MMC_1]
DEFAULT_logChanSet = Station_1
DEFAULT_logChan = N1_K1
ShowChanMenu = True
logChanSetList = Station_1, Station_2
[Station_1]
logChanList = N1_K1, N1_K2
[N1_K1]
logNCName = NCU_1
ChanNum = 1
[N1_K2]
logNCName = NCU_1
ChanNum = 2
 [Station_2]
logChanList = N1_K1, N1_K2
```

```
[N1_K1]
logNCName = NCU_2
ChanNum = 1
[N1_K2]
logNCName = NCU_2
ChanNum = 2
; End
```

Step 1b:

```
Create the NETNAMES.INI file for HT6
[own]
owner = HT_6
; Connection part
[conn HT_6]
conn_1 = NCU_2
; Network parameters
[param network]
bus= opi
; HMI descriptions
[param HT_6]
mmc_typ = 0x10
mmc_bustyp = OPI
mmc_address = 14
mstt_address = 14 ; is always the same as
; mmc address
name = MMC_Neben
start_mode = OFFLINE
; Description of NCU components
[param NCU_2]
type =NCU_573
nck_address = 23
plc_address = 23
name = NCU2
; Channel data
[chan HT_6]
DEFAULT_logChanSet = Station_2
```

DEFAULT_logChan = N1_K1

ShowChanMenu = True

logChanSetList = Station_2

[Station_2]

logChanList = N2_K1, N2_K2

[N2_K1]

logNCName = NCU_2

ChanNum = 1

[N2_K2]

logNCName = NCU_2

ChanNum = 2

;End of file

Step 2a:

PCU50:

Once the NETNAMES.INI file has been created, it is transferred or copied into the USER directory.

Step 2b:

HT6:

See example 2 for creation of the softkey texts.

After the NETNAMES.INI and chan.txt files have been created, they are included in the *.abb file with the application.

Step 3:

Set the NCK bus addresses

HMI Advanced/PCU50:

1. Enter the following in the "Operator panel front interface parameters" input screen via "Commissioning → HMI → Operator panel":
   Connection: M:N (Select instead of 1:1)
   NCK address: 22
   PLC address: 22
   according to NETNAMES.INI for NCU2 address 23
2. "Save"
3. Restart the PCU

Step 4:

Include FB9 in the PLC user program. You will find more details in the following section.

2.15.4.4 Description of FB9

Function description

This block allows **switchover** between several **operator panels** (PCU with operator panel and/or machine control panel), which are connected to one or more NCU control modules via a bus system.

The **interface** between the individual operator panels and the NCU (PLC) is the M:N interface in **data block DB19** (see Section "Signal descriptions" and /LIS/, Lists).

FB 9 uses the signals of this interface.

Apart from initialization, sign-of-life monitoring and error routines, the following **basic functions** are also performed by the block for control unit switchover:

Table 2- 7 Overview of the functions

Basic function	Meaning
PCU sends request	PCU wants to go online with an NCU
PCU coming	PCU is connecting to an NCU
PCU going	PCU is disconnecting from an NCU
Suppression	PCU must break connection with an NCU
Operating focus switchover in server mode	Switch operating focus from one NCU to the other
Active/passive operating mode	Operator control and monitoring/monitoring only
MCP switchover	As an option, MCP can be switched over with the PCU

The following descriptions supplement the information in "Description of operational sequences (overview)" and "Description of operational sequences (details)" with particular reference to the behavior in the last three examples.

Brief description of important functions

Active/passive operating mode

An online PCU can operate in two different modes:

Active mode: Operator can control and monitor

Passive mode: Operator can monitor (PCU header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two PCUs are connected online on one NCU simultaneously, one of the two is always in active and the other in passive mode. The operator can request active mode on the passive PCU at the press of a button.

MCP switchover:

As an option, an MCP assigned to the PCU can be switched over at the same time. This can be done on condition that the MCP address is entered in parameter mstt_adress of PCU configuration file NETNAMES.INI and MCPEnable is set to true. The MCP of the passive PCU is deactivated. so that there is only ever one active MCP on an NCU at one time.

Power-up condition:

To prevent the previously selected MCP being reactivated when the NCU is restarted, input parameters MCP1BusAdr = 255 (address of 1st MCP) and MCP1STOP =TRUE (deactivate 1st MCP) must be set when FB1 is called in OB 100.

Enabling commands:

When one MCP is switched over to another, any active feedrate or axis enabling signals may be transferred at the same time.

NOTICE
Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the HMI, which is subsequently activated). The override settings for feedrate and spindle also remain valid. To deactivate actuated keys, the input image of the machine control signals must be switched to non-actuated signal level on a falling edge of DB10.DBX104.0. The override settings should remain unchanged.

Measures for deactivating keys must be implemented in the PLC user program. (see below: Example of override switchover)

Declaration of the function

FUNCTION_BLOCK FB9

VAR_INPUT

Ack : BOOL; //Acknowledge alarms

OPMixedMode: BOOL:= FALSE ; // Mixed operation with non M:N-capable

// OP deactivated!

AktivEnable : BOOL:= TRUE ; // Activate active/passive switchover.

MCPEnable : BOOL:= TRUE ; // Activate MCP switchover

END_VAR

VAR_OUTPUT

Alarm1 : BOOL ; // Alarm: Error in PCU bus address, bus type!

Alarm2 : BOOL ; // Alarm: No confirmation MMC1 offline!

Alarm3 : BOOL ; // Alarm: MMC1 is not going offline!

Alarm4 : BOOL ; // Alarm: No confirmation MMC2 offline!

Alarm5 : BOOL ; // Alarm: MMC2 is not going offline!

Alarm6 : BOOL ; // Alarm: Requesting PCU is not going online!

Report : BOOL ;// Message: Sign-of-life monitoring

ErrorMMC : BOOL ; // Error detection HMI

END_VAR

Explanation of the formal parameters

The following table shows all formal parameters of function FB9

Table 2- 8 Formal parameters of FB9

Signal	Type	Type	Comment
Ack	I	BOOL	Acknowledge alarms
OPMixedMode	I	BOOL	Hybrid operation with non M:N-capable OP
AktivEnable	I	BOOL	Activate operator panel active/passive switchover TRUE = Operator panel can be switched to active/passive. FALSE = Operator panel cannot be switched to active and remains in its current state.
MCPEnable	I	BOOL	Activate MCP switchover TRUE = MCP is switched over with operator panel FALSE: = MCP is not switched over with operator panel.
Alarm1	O	BOOL	Alarm: Error in PCU bus address, bus type!
Alarm2	O	BOOL	Alarm: No confirmation PCU1 offline!
Alarm3	O	BOOL	Alarm: PCU1 is not going offline!
Alarm4	O	BOOL	Alarm: No confirmation PCU2 offline!
Alarm5	O	BOOL	Alarm: PCU2 is not going offline!
Alarm6	O	BOOL	Alarm: Requesting PCU is not going online!
Report	O	BOOL	Message: Sign-of-life monitoring
ErrorMMC	O	BOOL	Error detection HMI

Note

The user program must call the block. The user must provide an instance DB with any number for this purpose. The call is not multi-instance-capable.

2.15.4.5 Example of calling FB9

CALL FB 9 , DB 109 (

Ack := Fehler_Quitt, // e.g. MCP reset

OPMixedMode := FALSE,

AktivEnable := TRUE, // Enable PCU switchover

MCPEnable := TRUE, // Enable MCP switchover

Alarm1 := DB2.dbx188.0, // Error message 700.100

Alarm2 := DB2.dbx188.1, // Error message 700.101

Alarm3 := DB2.dbx188.2, // Error message 700.102

Alarm3 := DB2.dbx188.3, // Error message 700.103

Alarm3 := DB2.dbx188.4, // Error message 700.104

Alarm6 := DB2.dbx188.5, // Error message 700.105

Report := DB2.dbx192.0, // Operational message 700.132

ErrorMMC := DB2.dbx192.1) // Operational message 700.133

Note

AktivEnable := true enables PCU active/passive switchover.

MCPEnable := true allows MCP switchover.

The default value of this parameter is thus enabled and does not have to be enabled explicitly when the function is called.

Alarms, errors

The output parameters "Alarm1" to "Alarm6" and "Report" can be transferred to the DB2 areas for MMC alarm and error messages of the HMI software.

If execution of an HMI function has failed (and an appropriate error message cannot be displayed), status parameter ErrorMMC is set to 'logic 1' (e.g. error on power up when no connection is made).

Example of calling FB1

(call in OB100):

CALL "RUN_UP" , "gp_par" (

MCPNum := 1,

MCP1In := P#E 0.0,

MCP1Out := P#A 0.0,

MCP1StatSend := P#A 8.0,

MCP1StatRec := P#A 12.0,

MCP1BusAdr := 255, // Address of 1st MCP

MCP1Timeout := S5T#700MS,

MCP1Cycl := S5T#200MS,

MCP1Stop := TRUE, // MCP disabled

NCCyclTimeout := S5T#200MS,

NCRunupTimeout := S5T#50S);

2.15.4.6 Example of override switchover

The example uses auxiliary flags M100.0, M100.1, M100.2, M100.3.

The positive edge of MCP1Ready must check for override and initiate measures for the activation of the MCP block.

This example applies to the feedrate override. The interface and input bytes must be exchanged for spindle override.

```
U DB10.DBX 104.0; // MCP1Ready
 FN M 100.0; // Edge flag 1
 SPBN wei1;
 S M 100.2; // Set auxiliary flag 1
 R M 100.3; // Reset auxiliary flag 2
// Save override
 L DB21.DBB 4; // Interface feedrate override
 T EB 28; // Buffer storage (free input
// or flag byte)
wei1:
 U M 100.2; //Switchover has taken place
 O DB10.DBX 104.0; //MCP1Ready
 SPBN wei2;
 U DB10.DBX 104.0; // MCP1Ready
 FP M 100.1; // Edge flag 2
 SPB wei2;
 U M 100.2; //Switchover has taken place
 R M 100.2; // Reset auxiliary flag 1
 SPB wei2;
 U M 100.3; //Comparison has taken place
 SPB MCP; //Call MCP program
// Guide the stored override to the interface of the switched MCP
// until the override values match
 L EB28; //Buffer storage open
 T DB21.DBB 4; // Guide override interface
 L EB 3; //Override input byte for feed
 <>i; //Match?
 SPB wei2; //No, jump
 S M100.3; //Yes, set auxiliary flag 2
// When override values match, call the MCP program again
```

```
MCP: CALL "MCP_IFM"( //FC 19
 BAGNo := B#16#1,
 ChanNo := B#16#1,
 SpindleIFNo := B#16#0,
 FeedHold := M 101.0,
 SpindleHold := M 101.1);
wei2: NOP 0;
```

2.15.4.7 Switchover between MCP and HT6

```
CALL FCxx
 L DB7.DBB 27 // act. MCP
 L 6 // Machine control panel
 ==I
 SPB MSTT // Call FC 19
 L DB7.DBB 27 // act. MCP
 L 14 // HT 6
 ==I
 SPB HT6 // Call FC 26
 SPA END
HT6: NOP 0
 L B#16#40 // Shift the inputs of HT6 to IB 8+n
 T DB7.DBB7
 L B#16#40 // Shift the outputs of HT6 to OB 8+n
 T DB7.DBB13
 CALL FC26 // Block call HT6
 SPA END
MCP: NOP 0
 L 0
 T DB7.DBB7
 T DB7.DBB13

 CALL FC19 // Block call machine control panel
END: NOP 0
```

2.15.4.8 General Information

- In a configuration with only **one** NCU, the additional entry: " ,SAP=202 " must be set for the PLC address in the [param NCU_xx] section of the NETNAMES.INI file.

Example:

[param NCU_1]

type =NCU_573

nck_address = 11

plc_address= 11**, SAP=202**

name = NCU1

- In the event of a configuration without a machine control panel (operator panel without machine control panel), "mstt_address = **255**" must be entered in the parameter set in the [param MMC_xx] section of the relevant NETNAMES.INI file.
- This does not apply to HMI Embedded/HT6,as bt_conf outputs an error here.
- FB1 is configured by default in the PLC program (OB100 call), see "FB9 description".

Example:

[param MMC_1]

mmc_typ = 0x40

mmc_bustyp = OPI

mmc_address = 1

mstt_address = **255**

name = MMC_Serv

start_mode = ONLINE

- Recommendation: The OPI/MPI addresses 0 (for PG) and 13 (service case: NC replacement) should be kept free.
- OFFLINE mode for HMI Advanced: A server cannot be configured with boot property start_mode = Offline.

  If a main or secondary control panel is to be booted in offline mode, the following setting should be entered in the MMC.INI file.

  In the [Global] section, set

  NcddeDefaultMachineName = LOCAL.

  After you do this, you should not select "Save" in the "Operator panel front interface parameters" menu, otherwise this entry will be overwritten again.

HT6 removal/insertion

Trouble-free removal and insertion of the HT 6 **during machine operation** requires the following:

- Release or override of the HT 6 EMERGENCY STOP
- Connection of the HT 6 to the OPI/MPI via a PROFIBUS repeater.

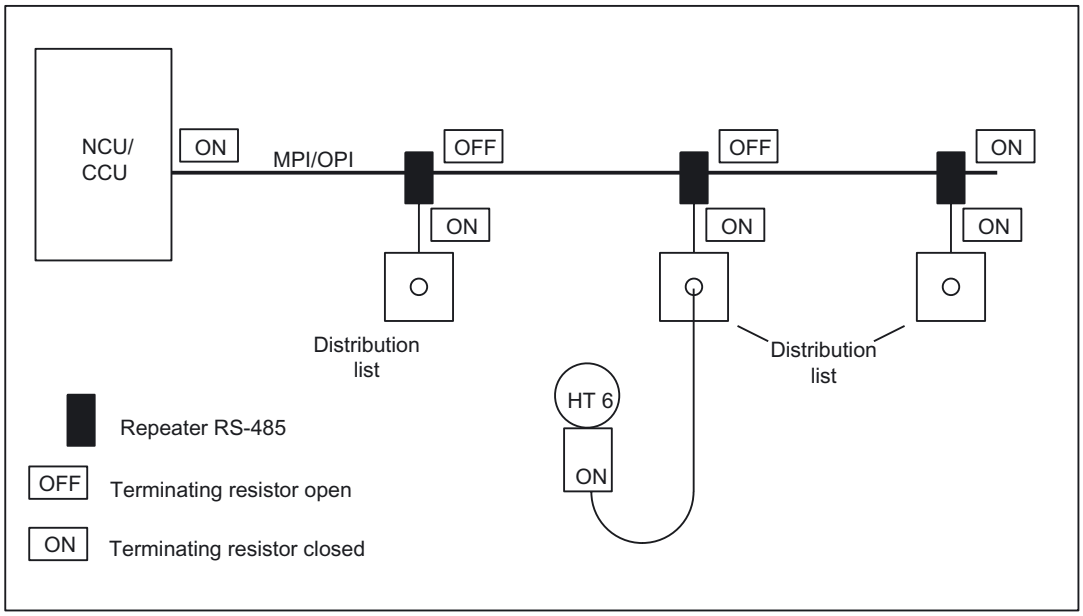


Figure 2-48 Connecting the HT 6 using a PROFIBUS repeater

A PROFIBUS repeater must be connected upstream of the HT 6 distributor box for each branch. The individual bus segments (MPI/OPI cable and/or the local segments between repeater and HT 6) must be terminated with connector resistors at the ends of the bus.

Repeater RS-485

The repeater can be ordered under Order No. 6ES7972-0AA01-0XA0. For further information, please refer to the Catalog

/IK10/ Industrial Communication Networks SIMATIC-NET

Note

Please note:

- The HT 6 already has an installed bus terminating resistor.
- The cable length from the repeater to the distributor box must not exceed 2 m.

You can find suggested circuits for the emergency stop in:

References: /BH/, Operator Components Manual.

2.15.5 Link axis

Assumption

NCU1 and NCU2 have one link axis each,
machine data e.g.:

```
; Machine data for NCU1:
$MN_NCU_LINKNO = 1 ; Set NCU number to 1
 ; (Master NCU)
$MN_MM_NCU_LINK_MASK = 1 ; Set link function to active
$MN_MM_SERVO_FIFO_SIZE = 3 ; Size of data buffer 1)
 ; between interpolation
 ; and position control
$MN_MM_LINK_NUM_OF_MODULES = 2 ; Number of link modules

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3" ; Link axis

; Unique NCU axis names
$MN_AXCONF_MACHAX_NAME_TAB[0] = "NC1_A1"
$MN_AXCONF_MACHAX_NAME_TAB[1] = "NC1_A2"
$MN_AXCONF_MACHAX_NAME_TAB[2] = "NC1_A3"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0] = 1
$MC_AXCONF_MACHAX_USED[1] = 2
$MC_AXCONF_MACHAX_USED[2] = 3

...

; Machine data for NCU2:
$MN_NCU_LINKNO = 2 ; Set NCU number to 2 (slave NCU)
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_SERVO_FIFO_SIZE = 3 ; 1)
$MN_MM_LINK_NUM_OF_MODULES = 2
```

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"

$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"

$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC1_AX3" ; Link axis

; Unique NCU axis names

$MN_AXCONF_MACHAX_NAME_TAB[0] = "NC2_A1"

$MN_AXCONF_MACHAX_NAME_TAB[1] = "NC2_A2"

$MN_AXCONF_MACHAX_NAME_TAB[2] = "NC2_A3"

CHANDATA(1)

$MC_AXCONF_MACHAX_USED[0] = 1

$MC_AXCONF_MACHAX_USED[1] = 2

$MC_AXCONF_MACHAX_USED[2] = 3

...

1) With software version 5 the machine data is:

MD10087 $MN_SERVO_FIFO_SIZE.

2.15.6 Axis container coordination

The characteristic as a function of time is displayed from top to bottom in the following tables. The data are valid on condition that only two channels have axes in the container.

2.15.6.1 Axis container rotation without a part program wait

Channel 1	Channel 2	Comment
AXCTWE(C1)	Part program ...	Channel 1 enables the axis container for rotation.
Part program **without** movement of a container axis	Part program ...	
	AXCTSWE(C1)	Channel 2 enables the axis container for rotation, container rotates because both channels have enabled rotation
Part program **with** movement of a container axis	Part program **with** movement of a container axis	**Without wait**

2.15.6.2 Axis container rotation with an implicit part program wait

Channel 1	Channel 2	Comment
AXCTWE(C1)	Part program ...	Channel 1 enables the axis container for rotation.
Part program **with** movement of a container axis	Part program ...	Channel 1 **waits implicitly** for axis container rotation
	AXCTSWE(C1)	Channel 2 enables the axis container for rotation, rotation occurs. Channel 1 is continued.

2.15.6.3 Axis container rotation by one channel only (e.g. during power up)

Channel 1	Channel 2	Comment
AXCTWE**D**(C1)	In the RESET state	Instantaneous rotation

2.15.7 Evaluating axis container system variables

2.15.7.1 Conditional branch

Channel 1	Comment
AXCTWE(CT1)	Channel 1 enables the axis container for rotation.
MARKER1: Part program **without** movement of a container axis	
IF $AC_AXCTSWA[CT1] == 1 GOTOB MARKE1	**Conditional branch** dependent on completion of axis container rotation.
Part program **with** movement of a container axis	

2.15.7.2 Static synchronized action with $AN_AXCTSWA

Channel 1	Comment
IDS =1 EVERY $AN_AXCTSWA[CT1] == 1 DO M99	Static synchronized action: Always output auxiliary function M99 at the beginning of an axis container rotation.
	References: /FBSY/, Function Manual Synchronized Actions

2.15.7.3 Wait for certain completion of axis container rotation

If you want to wait until the axis container rotation is reliably completed, you can use one of the examples below selected to suit the relevant situation.

Example 1

```
rl = $AN_AXCTAS[ctl]; Read current axis container position
AXCTSWE(ctl) ; Permit axis container rotation
WHILE (rl == $AN_AXCTAS[ctl]); Wait until axis container position
ENDWHILE ; has changed
```

Example 2 for 1st channel

```
CLEARM(9); Delete synchronization marker 9
AXCTSWE(ctl) ; Permit axis container rotation
; wait with synchronized action until
; axis container rotation is completed
WHEN $AN_AXCTSWA[ctl] == TRUE DO SETM(9) ; Set marker 9 and
WAITMC(9, 1) ; Wait for synchronization marker 9
; in first channel
```

Example 3.1 Use internal wait

```
M3 S100 ; Reprogram axis container spindle
; An internal wait takes place for the end of
; axis container rotation
```

Example 3.2 Use internal wait

```
x=IC(0) ; Reprogram axis container axis x
; An internal wait takes place for the end of
; axis container rotation
```

Example 3.3 Use internal wait

```
AXCTSWE(CTL) ; If an axis container is reenabled for rotation,
; an internal wait takes place for the end of the earlier
; axis container rotation.
```

N2150 WHILE (rl == $AN_AXCTAS[ctl])

Note

Programming in the NC program:

WHILE ($AN_AXCTSWA[n] == 0)

ENDWHILE

cannot be used as a reliable method of determining whether an earlier axis container rotation has finished. Although in software version 7.x and later, $AN_AXCTSWA performs an implicit preprocessing stop, this type of programming cannot be used, as the block can be interrupted by a reorganization. The system variable then returns "0" as the axis container rotation is then ended.

2.15.8 Configuration of a multi-spindle turning machine

Introduction

The following example describes the use of:

- Several NCUs in the NCU link group
- Flexible configuration with axis containers

Machine description

- Distributed on the circumference of a drum A (front-plane machining) the machine has:
  - 4 main spindles, HS1 to HS4

    Each main spindle has the possibility of material feed (bars, hydraulic bar feed, axes: STN1 - STN4).
  - 4 cross slides
  - Each slide has two axes.
  - Optionally a powered tool S1-S4 can operate on each slide.
- Distributed on the circumference of a drum B (rear-plane machining) the machine has:
  - 4 counterspindles GS1 to GS4
  - 4 cross slides
  - Each slide has two axes.
  - Optionally a powered tool S5-S8 can operate on each slide.
  - The position of each counterspindle can be offset through a linear axis for example for transferring parts from the main spindle for rear-plane machining in drum B. (Transfer axes. Axes: ZG1 - ZG4).
- Couplings:
  - If drum A rotates, **all** main spindles of this drum are subordinate to another group of slides.
  - If drum B rotates, **all** main counterspindles and all transfer axes of this drum are subordinate to another group of slides.
  - The rotations of drums A and B are autonomous.
  - The rotations of drums A and B are limited to 270 degrees.

    (range and twisting of supply cables)

Term: Position

Main spindle HS_i and counterspindle GS_i together with their slides characterize a position.

NCU assignment

The axes and spindles of a position (for this example) are each assigned to an NCU. One of the NCUs, the master NCU, controls the axes for the rotations of drums A and B additionally. There are 4 NCUs with a maximum of the following axes:

Number of axes

Per NCU_i the following axes/spindles must be configured:

Slide 1: X_i1, Z_i1

2: X_i2, Z_i2

Spindles: HS_i, GS_i, powered tools: S1, S2

Transfer axis: ZG_i

Bar feed: STN_i.

For the master NCU, in addition to the above-mentioned axes there are the two axes for rotating drums A and B. The list shows that it would not be possible to configure the axis number for a total of 4 positions via an NCU. (Limit 31 axes, required are 4 + 10 + 2 axes).

Axis container

With rotation of drums A/B, HS_i, GS_i, ZG_i and STN_i must be assigned to another NCU and must therefore be configured as link axes in axis containers.

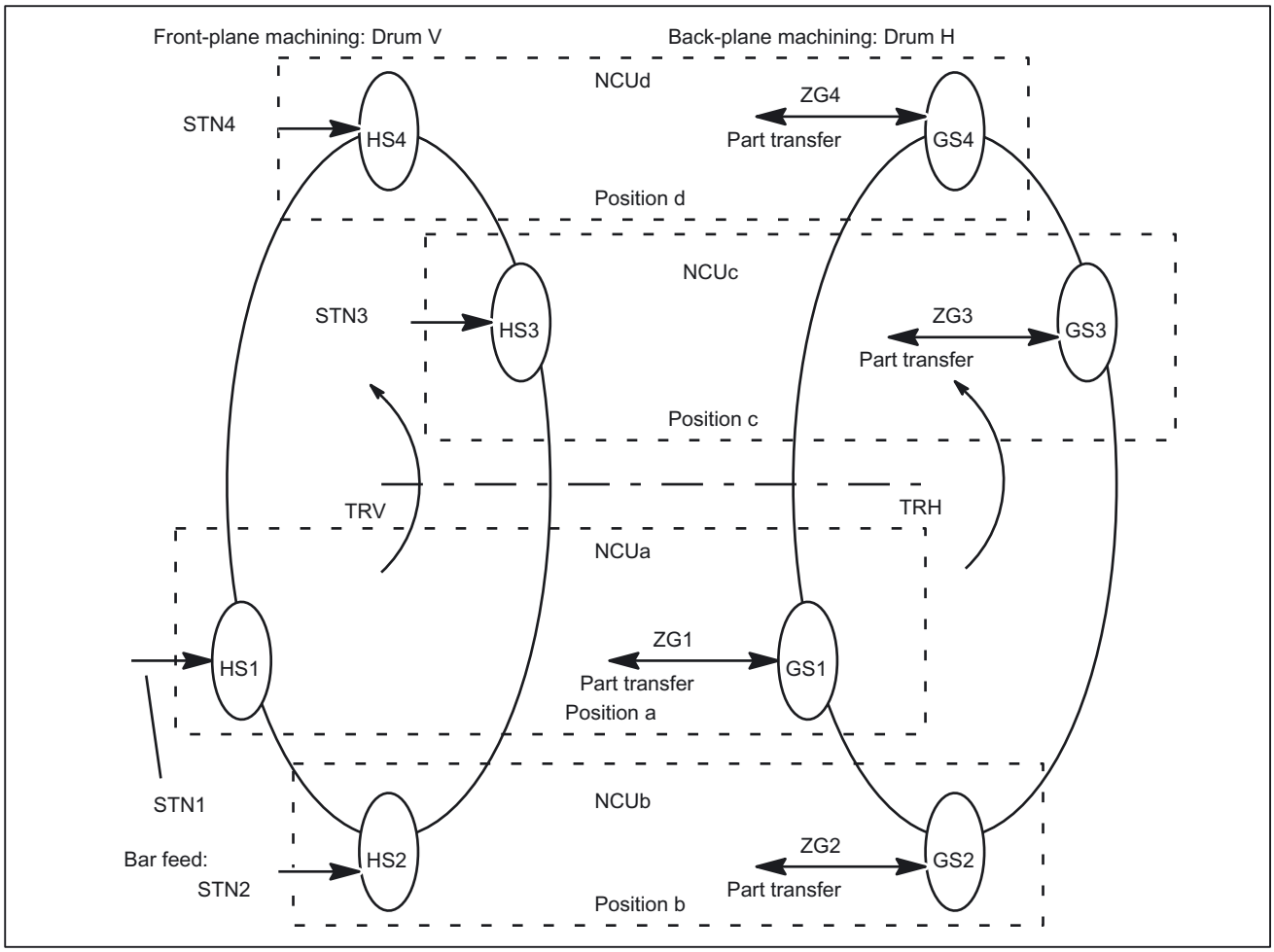


Figure 2-49 Schematic diagram of main spindles HSi, countersp. GSi, bar feed axis STNi and transfer axes ZGi

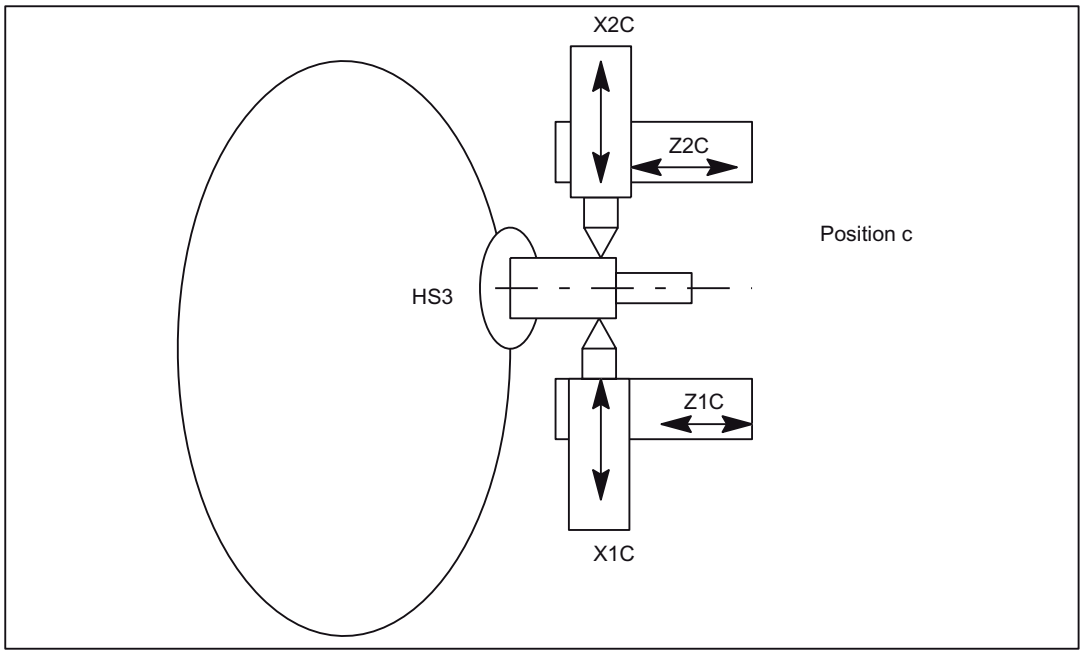


Figure 2-50 Two slides per position can also operate together on one spindle

Note

The axes are given the following names in order to clarify the assignments of axes to slides and positions:

Xij with i slide (1, 2), j position (A-D)

Zij with i slide (1, 2), j position (A-D)

Whereas the positions and their slides remain in one place, main spindles, counterspindles, bar feed axes STN and transfer axes ZG move to new positions as the result of rotations of drums V or H.

For example, the axes to be managed per NC when the slide is taken into account are as follows for the configurations shown in the above diagrams:

Axes of master NCU

Table 2- 9 Axes of master NCU: NCUa

Common axes	Local axes	Comment
	TRV (drum V)	Master NCU only
	TRH (drum H)	Master NCU only
	X1A	Slide 1
	Z1A	Slide 1
	X2A	Slide 2

Common axes	Local axes	Comment
	Z2A	Slide 2
	S1	Slide 1
	S2	Slide 2
HS1		Axis container necessary
GS1		Axis container necessary
ZG1		Axis container necessary
STN1		Axis container necessary
4	8	

Axes of NCUb to NCUd

The NCUs that are not master NCUs have the same axes with the exception of the axes for the drive for drums TRV and TRH. The letter designating the position must be replaced accordingly for the NCU and axis name (a, A → b, B to d, D).

Configuration rules

The following rules were applied for the configuration described below:

- Main spindle, counterspindles and axes that are assigned to different NCUs through drum rotation while they are operating as illustrated in the above diagram "Main spindle ..." must be configured in an axis container.

  (HS_i, GS_i, ZG_i, STN_i).
- All main spindles for drum A are in the same container (No. 1).
- All bar feed axes for drum A are in the same container (No. 2).
- All counterspindles for drum B are in the same container (No. 3).
- All transfer axes for drum B are in the same container (No. 4).
- Main spindles HS_i and their counterspindle GS_i as well as the transfer axes for counterspindle ZG_i and the bar feed axes STN_i of the main spindle are assigned as follows for uniform load distribution purposes:

  NCUa HS1-STN1,

  NCUb HS2-STN2, ... etc.
- Slide axes Xij, Zij are solely local axes with a fixed NCU assignment.
- Slides are assigned to a dedicated channel of an NCU.

  Slides can therefore be moved autonomously.

Configuration options

- Main or counterspindles are flexibly assigned to the slide.
- The speed of the main spindle and the counterspindle can be defined independently in each position.

  Exceptions:

  During the parts change from front-plane machining in drum V to rear-plane machining in drum H, the speeds of the main spindle and the counterspindle must be synchronized (synchronous spindle coupling).

  In cases where slide 2 also participates in front-plane machining so as to "support" slide 1, the main spindle speed also applies to slide 2. Similarly if slide 1 participates in rear-plane machining, the counterspindle speed also applies to slide 1.

Minor changes in speed

Due to the unavoidable time delays incurred in the processing of actual values, abrupt changes in speed should be avoided during cross-NCU machining operations. Compare axis data and signals.

Configuration for NCU1

Uniform use of channel axis names in the part programs:

S4: Main spindle

S3: Counterspindle

X1: Infeed axis

Z1: Longitudinal axis

S1: Powered tool

Z3: Transfer axis

TRV: Drum V for main spindle

TRH: Drum H for counterspindle

STN: Hydraulic bar feed

Axes highlighted in **bold** characterize the current channel as home channel for the axis in conjunction with axis replacement.

Table 2- 10 NCUa, position: a, channel: 1, slide: 1

Channel axis name	**..._MACHAX_USED**	**$MN_AXCONF_LOGIC_MACHAX_TAB,**	**Container, slot entry (string)**	**Machine axis name**
S4	1	AX1: CT1_SL1	1 1 NC1_AX1	HS1
S3	2	AX2: CT3_SL1	3 1 NC1_AX2	GS1
X1	3	AX3:		X1A

Channel axis name	..._MACHAX _USED	$MN_ AXCONF_LOGIC_MACH AX_TAB,	Container, slot entry (string)	Machine axis name
Z1	4	AX4:		Z1A
Z3	5	AX5: CT4_SL1	4 1 NC1_AX5	ZG1
S1	*6*	*AX6:*		*WZ1A*
STN	7	AX7: CT2_SL1	2 1 NC1_AX7	STN1
TRV	11	AX11:		TRV
TRH	12	AX12:		TRH
x2 *				
z2 *				

Table 2- 11 NCUa, position: a, channel: 2, slide: 2

Channel axis name	..._MACHAX _USED	$MN_ AXCONF_LOGIC_MACH AX_TAB,	Container, slot entry (string)	Machine axis name
S4	1	AX1: CT1_SL1	1 1 NC1_AX1	HS1
S3	2	AX2: CT3_SL1	3 1 NC1_AX2	GS1
Z3	5	AX5: CT4_SL1	4 1 NC1_AX5	ZG1
STN	7	AX7: CT2_SL1	2 1 NC1_AX7	STN1
X2	8	AX8:		X2A
Z2	9	AX9:		Z2A
S1	*10*	*AX10:*		*WZ2A*
x1 *				
z1 *				

Note

* due to program coordination via axis positions and 4-axis machining in one position

Entries in the axis container locations should have the following format: "NC1_AX.." with the meaning NC1 = NCU 1. In the above tables, NCUa is imaged on NC1_..., NCUb on NC2_... etc.

Further NCUs

The above listed configuration data must be specified accordingly for NCUb to NCUd. Please note the following:

- Axes TRA and TRB only exist for NCUa, channel 1.
- The container numbers are maintained for the other NCUs as they were specified for the individual axes
- The slot numbers are as follows:

  NCUb → 2

  NCUc → 3

  NCUd → 4.
- The machine axis names are as follows:

  NCUb → HS2, GS2, ZG2, STN2

  NCUc → HS3, GS3, ZG3, STN3

  NCUd → HS4, GS4, ZG4, STN4.

Axis container

The information relating to containers given in Table 7-17 and the container entries of the similarly configured NCUs, NCUb to NCUd, are specified in the following tables, sorted according to containers and slots, as they have to be set in machine data:

MD12701 $MN_AXCT_AXCONF_ASSIGN_TAB1[slot]

. . .

MD12716 $MN_AXCT_AXCONF_ASSIGN_TAB16[slot]

whereby slots: 1-4 must be set for the 4 positions of a multi-spindle turning machine:

Note

For the machine data entry

$MN_AXC_AXCONF_ASSIGN_TAB$_i$[slot]

the values (without decimal point and machine axis name) that are entered under initial position in the above tables must be set.

Table 2- 12 Axis container and their position-dependent contents for drum A

Container	Slot	Initial position (TRA 0°)	Switch 1 (TRA 90°)	Switch 2 (TRA 180°)	Switch 3 (TRA 270°)	Switch 4 = (TRA 0°)
1	1	NC1_AX1, HS1	NC2_AX1, HS2	NC3_AX1, HS3	NC4_AX1, HS4	NC1_AX1, HS1
	2	NC2_AX1, HS2	NC3_AX1, HS3	N4C_AX1, HS4	NC1_AX1, HS1	NC2_AX1, HS2
	3	NC3_AX1, HS3	NC4_AX1, HS4	NC1_AX1, HS1	NC2_AX1, HS2	NC3_AX1, HS3
	4	NC4_AX1, HS4	NC1_AX1, HS1	NC2_AX1, HS2	NC3_AX1, HS3	NC4_AX1, HS4

Container	Slot	Initial position (TRA 0°)	Switch 1 (TRA 90°)	Switch 2 (TRA 180°)	Switch 3 (TRA 270°)	Switch 4 = (TRA 0°)
2	1	NC1_AX7, STN1	NC2_AX7, STN2	NC3_AX7, STN3	NC4_AX7 STN4	NC1_AX7, STN1
	2	NC2_AX7, STN2	NC3_AX7, STN3	NC4_AX7, STN4	NC1_AX7, STN1	NC2_AX7, STN2
	3	NC3_AX7, STN3	NC4_AX7, STN4	NC1_AX7, STN1	NC2_AX7, STN2	NC3_AX7, STN3
	4	NC4_AX7, STN4	NC1_AX7, STN1	NC2_AX7, STN2	NC3_AX7, STN3	NC4_AX7, STN4
Drum movement		0°	+ 90°	+ 90°	+ 90°	- 270°

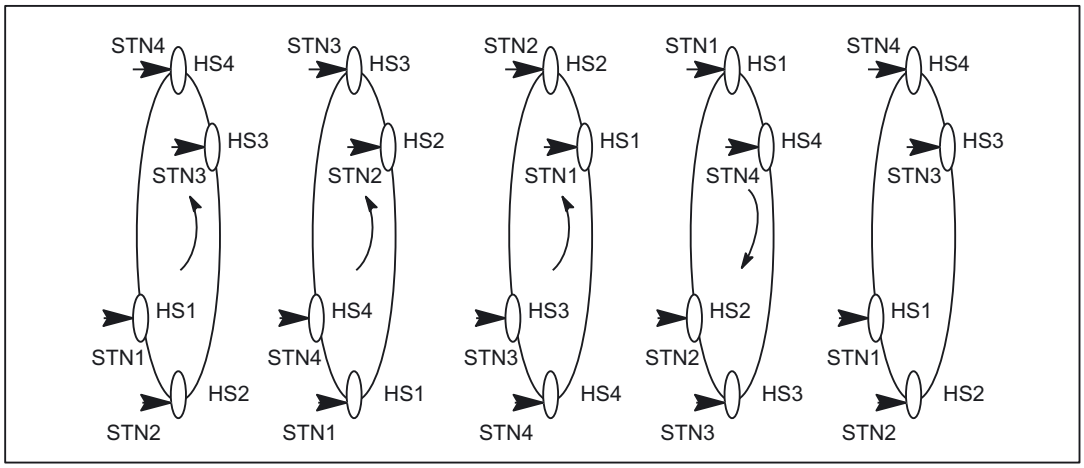


Figure 2-51 Positions of drum A

Table 2- 13 Axis container and their position-dependent contents for drum B

Container	Slot	Initial position (TRB 0°)	Switch 1 (TRB 90°)	Switch 2 (TRB 180°)	Switch 3 (TRB 270°)	Switch 4 = (TRB 0°)
3	1	NC1_AX2, GS1	NC2_AX2, GS2	NC3_AX2, GS3	NC4_AX2, GS4	NC1_AX2, GS1
	2	NC2_AX2, GS2	NC3_AX2, GS3	NC4_AX2, GS4	NC1_AX2, GS1	NC2_AX2, GS2
	3	NC3_AX2, GS3	NC4_AX2, GS4	NC1_AX2, GS1	NC2_AX2, GS2	NC3_AX2, GS3
	4	NC4_AX2, GS4	NC1_AX2, GS1	NC2_AX2, GS2	NC3_AX2, GS3	NC4_AX2, GS4
4	1	NC1_AX5, ZG1	NC2_AX5, ZG2	NC3_AX5, ZG3	NC4_AX5 ZG4	NC1_AX5, ZG1
	2	NC2_AX5, ZG2	NC3_AX5, ZG3	NC4_AX5, ZG4	NC1_AX5, ZG1	NC2_AX5, ZG2
	3	NC3_AX5, ZG3	NC4_AX5, ZG4	NC1_AX5, ZG1	NC2_AX5, ZG2	NC3_AX5, ZG3
	4	NC4_AX5, ZG4	NC1_AX5, ZG1	NC2_AX5, ZG2	NC3_AX5, ZG3	NC4_AX5, ZG4

2.15.9 Lead link axis

2.15.9.1 Configuration

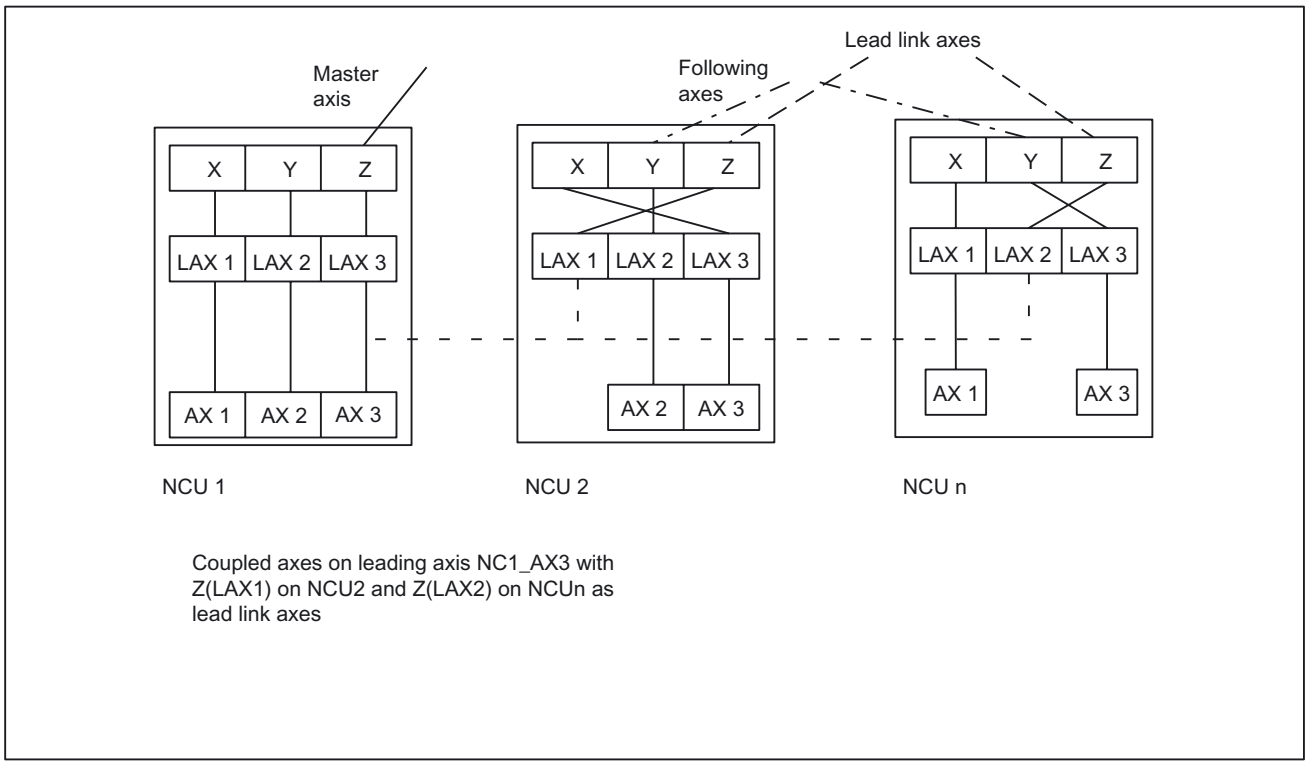


Figure 2-52 NCU2 to NCUn use a lead link axis to enable coupling to the machine axis on NCU1 (NCU1-AX3).

The following example refers to the axis coupling section between Y(LAX2, AX2) as following axis on NCU2 and Z(LAX3, NC1_AX3) as lead link axis.

Loading the machine data

1. The machine data/setting data of a master value axis may only be loaded on the home NCU. The machine data are distributed internally to the other NCUs where a lead link axis has been defined.
2. The lead link axis must be taken into account when configuring the NCU that is traversing the following axes (NCU2). The lead link axis occupies one location in the logical machine axis image (LAI) of the (NCU2). This reduces the maximum number of axes to be interpolated by this NCU by 1 for the lead link axis.

   In addition to the LAI axis location definition, the lead link axis must also be defined as channel axis ($MC_AXCONF_MACHAX_USED) in every channel where it will be used together with the following axis; this also reduces the maximum number of possible channel axes.

Machine data for NCU1

```
NCU traversing leading axis
$MN_NCU_LINKNO = 1 ; Master NCU
$MN_MM_NCU_LINK_MASK = 1 ; NCU link active
$MN_MM_LINK_NUM_OF_MODULES = 2 ; Number of link modules
$MN_MM_SERVO_FIFO_SIZE = 4 ; Size of data buffer
; increased to 4 between interpolation and position control
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "AX3"
$MA_AXCONF_ASSIGN_MASTER_NCK[ AX3 ] = 1
$MN_AXCONF_MACHAX_NAME_TAB[0] = "XM1"
$MN_AXCONF_MACHAX_NAME_TAB[2] = "YM1"
CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1 ; X
$MC_AXCONF_MACHAX_USED[1]=2 ; Y
$MC_AXCONF_MACHAX_USED[2]=3 ; Z
```

Machine data for NCU2

```
NCU(s) traversing following axis
$MN_NCU_LINKNO = 2 ; Set NCU number to 2
$MN_MM_NCU_LINK_MASK = 1 ; Activate link
$MN_MM_NUM_CURVE_TABS = 5 ; Number of curve tables
$MN_MM_LINK_NUM_OF_MODULES = 2 ; Number of link modules
$MN_MM_NUM_CURVE_SEGMENTS = 50
$MN_MM_NUM_CURVE_POLYNOMS = 100
$MN_MM_SERVO_FIFO_SIZE = 2 ; Size of data buffer
; between interpolation and position control (standard)
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC1_AX3" ; Lead link on
 ; NCU1/AX3
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "AX3
CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=3 ; X
$MC_AXCONF_MACHAX_USED[1]=2 ; Y
$MC_AXCONF_MACHAX_USED[2]=1 ; Z ; Assignment to LAI AX1
 ; or NCU1/AX3
```

2.15.9.2 Programming

Program on NCU 1

NCU1 traverses leading axis Z. The variable is 1 for as long as NCU2 is prepared for movement of the leading axis (messages via link variable $A_DLB[0]); after completion of movement, the variable is 0.

```
N3000 R1 = 1 ; Counter for movement loop
N3004 G1 Z0 F1000
N3005 $A_DLB[0] = 1 ; Start on NCU1
LOOP30:
N3005 R1=R1+1
N3006 G91 Z0.01 ; The master value axis is now traversed
N3008 Z0.02
N3010 Z0.03
N3012 IF R1 < 10 GOTOB LOOP30
N3098 $A_DLB[0] = 0 ; End on NCU1
N3099 GOTOF TESTE
```

NC program on NCU2

The program establishes a connection between leading axis movements on NCU1 and following axis movements on NCU2 via a curve table. Once the table has been defined, NCU2 goes to wait position until NCU1 starts the leading axis. Then the coupling is activated and maintained until the leading axis movement is terminated.

```
N2800 CTABDEL(1)
N2801 G04 F.1
N2803 G0 Y0 Z0
;*****************************************************
; Create table 1
;*****************************************************
N2802 CTABDEF(Y, Z, 1, 0)
N2803 G1 X0 Y0
N2804 G1 X100 Y200
N2805 CTABEND
LOOP29:
N2806 IF ($A_DLB[0]== 0) GOTOB LOOP29 ; Wait for NCU1
N2810 LEADON(Y,Z,1)
LOOP292: ; Coupling active!!!
N2830 IF ($A_DLB[0] > 0) GOTOB LOOP292 ; Maintain coupling until
; NCU1 is no longer traversing the master value axis
N2890 LEADOF(Y,Z)
```

2.15.10 NCU link with different interpolation cycles

2.15.10.1 Example of eccentric turning

Task assignment

Create a non-circular shape with the following characteristics:

Ellipticity: 0.2 mm

Base circle diameter: 50 mm

Z path per revolution: 0.1 mm

Spindle speed: 3000 rpm

A sinusoidal approximation via a cubic polynomial per 45 degrees of spindle revolution should be sufficient for the required precision.

Note

Polynomials up to the 5th degree can be used. See Programming Manual Job Planning.

The part program of which excerpts are shown runs in a channel of the NCU with the faster interpolation cycle where the X axis is configured as local axis and the C and Z axes as link axes.

CAD systems which calculate coefficients based on contour points and the required degree of the polynomial are used for defining the polynomial coefficients.

The following part program describes the commands required for the first spindle revolution. It must then be continued accordingly for the entire required length of the Z path:

```
G0 C0 X24.95 Z0 ; Start position
FGROUP(C) ; Ensures constant spindle speed
G1 G642 F1080000 ; Spindle speed 3000 rpm
POLY ;Specification of polynomials
C=DC(45.0000000) PO[X]=(25.0,.0750000,-0.0250000)
PO[Z]=(.2125000,0,0)
;1/8 circle, lineare Z movement, 1/4 sine in X
C=DC(90.0000000) PO[X]=(25.0500000,0,-0.0250000)
PO[Z]=(.2250000,0,0)
C=DC(135.0000000) PO[X]=(25.0,-0.0750000,.0250000)
PO[Z]=(.2375000,0,0)
C=DC(180.0000000) PO[X]=(24.9500000,0,.0250000) PO[Z]=(.2500000,0,0)
C=DC(225.0000000) PO[X]=(25.0,.0750000,-0.0250000)
PO[Z]=(.2625000,0,0)
C=DC(270.0000000) PO[X]=(25.0500000,0,-0.0250000)
PO[Z]=(.2750000,0,0)
C=DC(315.0000000) PO[X]=(25.0,-0.0750000,.0250000)
PO[Z]=(.2875000,0,0)
C=DC(0) PO[X]=(24.9500000,0,.0250000) PO[Z]=(.3000000,0,0)
...
```

Machine data

Please refer to "Configuration of link axes", "Axis container" for notes on the configuration of machine data.

2.16 Data lists

2.16.1 Machine data

2.16.1.1 General machine data

Number	Identifier: $MN_	Description
10002	AXCONF_LOGIC_MACHAX_TAB[n]	Logical NCU machine axis image
10065	POSCTRL_DESVAL_DELAY	Position setpoint delay
10087	SERVO_FIFO_SIZE	Size of data buffer between interpolation and position controller task (up to software version 5, then MD18720, see below)
10134	MM_NUM_MMC_UNITS	Number of simultaneous MMC communication partners
11398	AXIS_VAR_SERVER_SENSITIVE	Response of the AXIS-VAR server to errors
12510	NCU_LINKNO	NCU number in an NCU group
12520	LINK_TERMINATION	NCU numbers for which bus terminating resistors are active
12530	LINK_NUM_OF_MODULES	Number of NCU link modules
12540	LINK_BAUDRATE_SWITCH	Link bus baud rate
12550	LINK_RETRY_CTR	Maximum number of message frame repeats in event of error
12701 ... 12716	AXCT_AXCONF_ASSIGN_TAB1[s] ... AXCT_AXCONF_ASSIGN_TAB16[s]	List of axes in the axis container
12750	AXCT_NAME_TAB[n]	List of axis container names
18700	MM_SIZEOF_LINKVAR_DATA	Size of the NCU link variable memory
18720	MM_SERVO_FIFO_SIZE	Size of data buffer between interpolation and position control task
18780	MM_NCU_LINK_MASK	Activation of NCU link communication

2.16.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20000	CHAN_NAME	Channel name
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
28160	MM_NUM_LINKVAR_ELEMENTS	Number of write elements for the NCU link variables

2.16.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30550	AXCONF_ASSIGN_MASTER_CHAN	Default assignment between an axis and a channel
30554	AXCONF_ASSIGN_MASTER_NCU	Initial setting defining which NCU generates setpoints for the axis
30560	IS_LOCAL_LINK_AXIS	Axis is a local link axis
32990	POCTRL_DESVAL_DELAY_INFO	Current position setpoint delay

2.16.2 Setting data

2.16.2.1 General setting data

Number	Identifier: $SA	Description
41700	AXCT_SWWIDTH[container number]	Axis container rotation setting

2.16.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43300	ASSIGN_FEED_PER_REV_SOURCE	Rotational feedrate for positioning axes/spindles

2.16.3 Signals

2.16.3.1 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
MCP1 ready	DB10.DBX104.0	-
MCP2 ready	DB10.DBX104.1	-
HHU ready	DB10.DBX104.2	-
NCU link active	DB10.DBX107.6	-
HMI2-CPU ready (HMI connected to OPI or MPI)	DB10.DBX108.1	-
HMI1-CPU at MPI ready	DB10.DBX108.2	-
HMI1-CPU at OPI ready (standard connection)	DB10.DBX108.3	DB2700.DBX2.3

2.16.3.2 Signals from HMI/PLC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
ONL_REQUEST Online request from HMI	DB19.DBB100	-
ONL_CONFIRM Acknowledgement from PLC for online request	DB19.DBB102	-
PAR_CLIENT_IDENT HMI writes its client identification (bus type, HMI bus address)	DB19.DBB104	-
PAR_MMC_TYP HMI type according to NETNAMES.INI: Main/secondary operator panel/alarm server	DB19.DBB106	-
PAR_MSTT_ADR HMI writes address to the MCP to be activated	DB19.DBB107	-
PAR_STATUS PLC writes the online enable for the HMI (connection state)	DB19.DBB108	-
PAR_Z_INFO PLC writes additional information to the connection state	DB19.DBB109	-
M_TO_N_ALIVE Sign of life from the PLC to HMI using M to N block	DB19.DBB110	-

2.16.3.3 General online interface

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
MMC1_CLIENT_IDENT PLC writes PAR_CLIENT_IDENT to MMCx_CLIENT_IDENT, if HMI goes online.	DB19.DBB120	-
MMC1_TYP PLC writes PAR_MMC_TYP to MMCx_TYP, if HMI goes online.	DB19.DBB122	-
MMC1_MSTT_ADR PLC writes PAR_MSTT_ADR to MMCx_MSTT_ADR, if HMI goes online	DB19.DBB123	-
MMC1_STATUS Connection state, HMI and PLC write alternating, their requests/acknowledgements.	DB19.DBB124	-
MMC1_Z_INFO Additional information, connection state (pos./neg. acknowledgement, error messages, ...)	DB19.DBB125	-
MMC1_SHIFT_LOCK HMI switchover lock	DB19.DBX126.0	-

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
MMC1_MSTT_SHIFT_LOCK MCP shiftover lock	DB19.DBX126.1	-
MMC1_ACTIVE_REQ HMI requests active operator mode	DB19.DBX126.2	-
MMC1_ACTIVE_PERM Enable from PLC to change the operator mode	DB19.DBX126.3	-
MMC1_ACTIVE_CHANGED HMI has changed the operator mode	DB19.DBX126.4	-
MMC1_CHANGE_DENIED HMI active/passive switchover denied	DB19.DBX126.5	-
MMC2_CLIENT_IDENT PLC writes PAR_CLIENT_IDENT to MMCx_CLIENT_IDENT, if HMI goes online.	DB19.DBB130	-
MMC2_TYP PLC writes PAR_MMC_TYP to MMCx_TYP, if HMI goes online.	DB19.DBB132	-
MMC2_MSTT_ADR PLC writes PAR_MSTT_ADR to MMCx_MSTT_ADR, if HMI goes online.	DB19.DBB133	-
MMC2_STATUS Connection state, HMI and PLC write alternating, their requests/acknowledgements.	DB19.DBB134	-
MMC2_Z_INFO Additional information, connection state (pos./neg. acknowledgement, error messages, ...)	DB19.DBB135	-
MMC2_SHIFT_LOCK HMI switchover lock	DB19.DBX136.0	-
MMC2_MSTT_SHIFT_LOCK MCP shiftover lock	DB19.DBX136.1	-
MMC2_ACTIVE_REQ HMI requests active operator mode	DB19.DBX136.2	-
MMC2_ACTIVE_PERM Enable from PLC to change the operator mode	DB19.DBX136.3	-
MMC2_ACTIVE_CHANGED HMI has changed the operator mode	DB19.DBX136.4	-
MMC2_CHANGE_DENIED HMI active/passive switchover denied	DB19.DBX136.5	-

2.16.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NCU link axis active	DB31, ...DBX60.1	-
Axis container rotation active	DB31, ...DBX61.1	DB390x.DBX1.1
Axis ready	DB31, ...DBX61.2	DB390x.DBX1.2

B4: Operation via PG/PC - only 840D sl

3

3.1 Brief description

Applications

Operation via PG/PC

- must be utilized if no operator panel front is installed.
- can be utilized as a handling support for OP030 panels.

Hardware

The following hardware requirements must be fulfilled:

- PG/PC with at least 486DX33 processor and 8 MB main memory
- MS Windows must be running in ENHANCED mode (386 mode)
- PG/PC with MPI/OPI interface (implemented with PG 720/720C/740/760).

  An MPI card (6ES7793-2AA00-0AA0) can be supplied for PCs with a free ISA slot.
- VGA monitor with a resolution of 640x480 or higher.

Implementation Variant 1

The machine control panel (MCP) and operator panel OP030 are permanently allocated to the NCU.

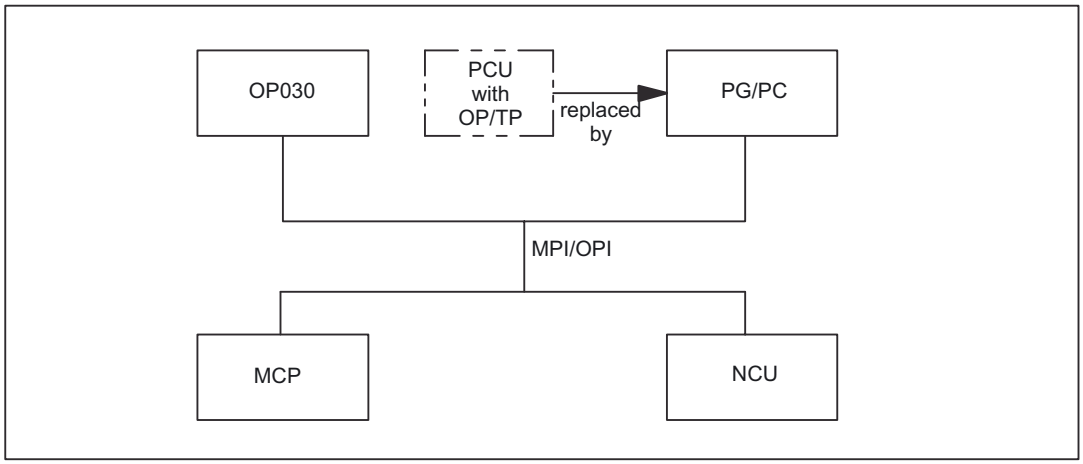


Figure 3-1 Configuration with OP030 and PG/PC

Machine control panel, OP030 and NCU are all connected either to the OPI bus or the MPI bus. A **homogenous** network must be provided with respect to these components.

Implementation Variant 2

Operator panel front and up to three NCUs The machine control panel is permanently allocated to the NCU concerned.

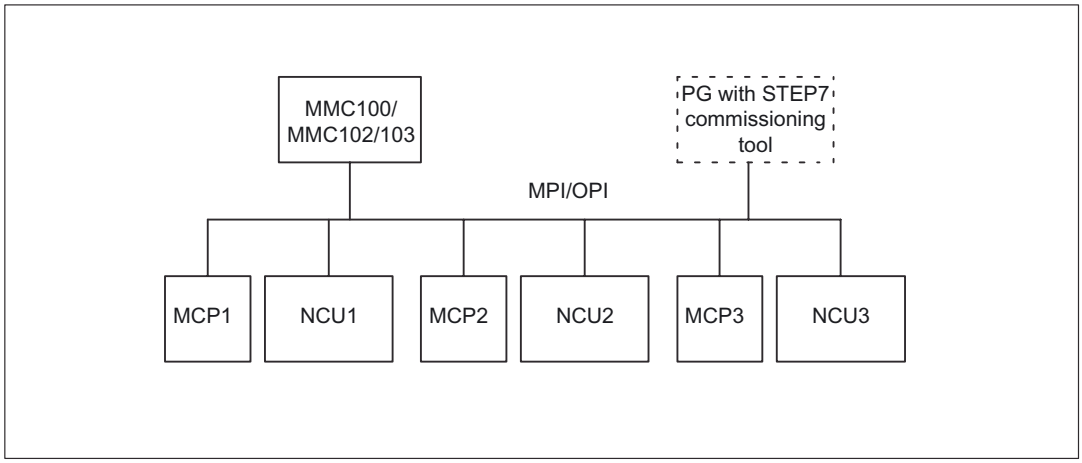


Figure 3-2 Configuration m:n corresponds to 1:3

Reference:

/FB2/ Function Manual, Extended Functions; Several Operator Panel Fronts and NCUs (B3)

Operator interfaces

The operator interfaces are described in the operating manuals for the operator panel fronts used.

Reference:

/BA/ Operating Manual

/FBO/ Configuration of Operator Interface OP 030; BA, Operating Manual OP030

Restrictions

If operation via PG/PC is used in addition to an OP030 operator panel front, the conditions relating to configuration and coordination described in /FB2/, B3, "Several operator panel fronts and NCUs" must be observed.

Reference:

/FB2/ Function Manual, Extended Functions; Several Operator Panel Fronts and NCUs (B3)

3.2 Software installation

3.2.1 System requirements

Hardware requirements

The following hardware requirements must be fulfilled to allow operation via PG/PC:

- IBM® AT-compatible PG/PC with 486DX33 microprocessor
- At least 8 MB of main memory
- Diskette drive (3½ inch)
- Hard disk drive for data management
- Monochrome or color monitor
- Keyboard
- PG/PC with MPI interface (implemented with PG 720/720C/740/760)

  Restricted operation is possible without an MPI card (e.g. interactive operation).

  Note: RS-232 MPI adapter is not supported.
- Mouse
- Connecting cable to link PG/PC and NCU module

Note

The operator panel fronts and the NCU are either

- all connected to the OPI bus or
- all connected to the MPI bus.

A **homogenous** network must be provided with respect to these components.

Software requirements

Software configuration for operation via PG/PC:

- MS-DOS operating system®, version 6.x or higher
- WINDOWS™ operator interface, version 3.1 or higher
- MPI interface driver (contained in the supplied software).
- WINDOWS™ 32s, version 1.30.166.0 or higher

  (You can check which version is installed under "windows\system\win32s.ini".)

  If WINDOWS™ 32s is not installed, you can use the 2 supplied diskettes to install it (run setup.exe).

3.2.2 Installation

Storage area of MPI card

The storage area of the MPI card must be excluded from use by the memory manager (files: CONFIG.SYS, SYSTEM.INI).

Example for entry in SYSTEM.INI:

[386enh]

EmmExclude=....<storage area of card>

(See hardware description of card)

Scope of delivery

System software:

- Approx.10 diskettes with compressed MMC102/103 software and installation tools
- 2 diskettes with WINDOWS 32s subsystem (= Microsoft setup).

To install the software, please follow the procedure detailed below:

Start

1. Run SETUP.EXE

Insert the first installation diskette and use the WINDOWS™ file manager to run SETUP.EXE.

The installation program requests all further necessary inputs and disk changes in user dialog.

2. Enter installation path

Select the directory plus the installation path (see screenshot) to which you wish to copy the software.

Press "**Continue**" to continue the installation or "**Exit Setup**" to interrupt the installation procedure.

This also applies to further operations.

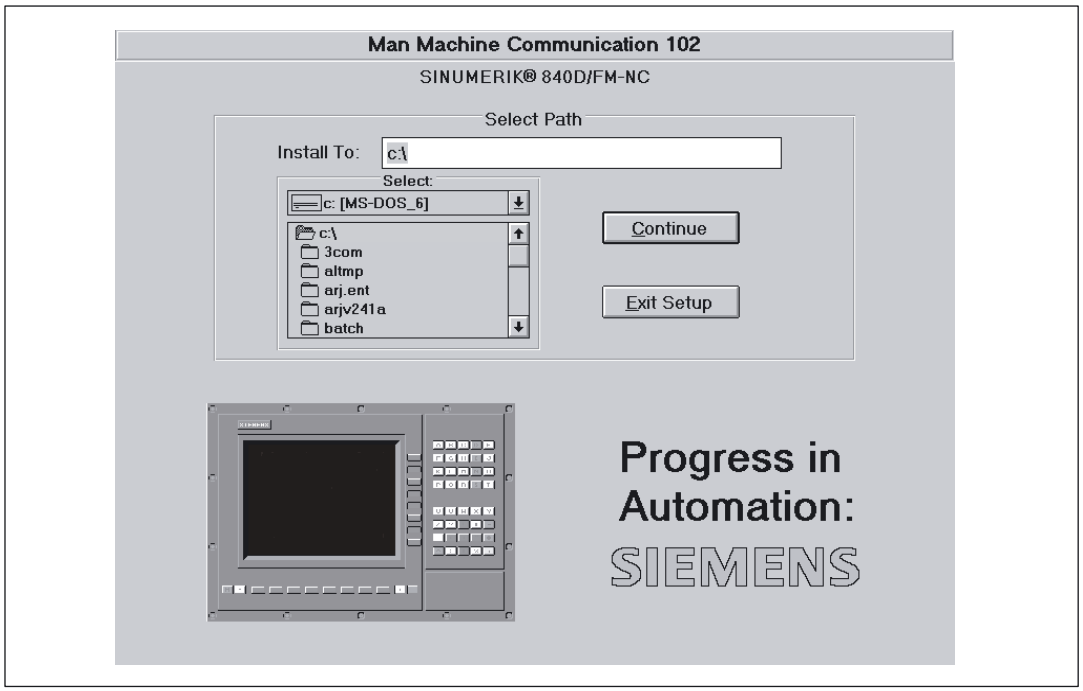


Figure 3-3 Enter installation path

3. Select operation with MPI or without MPI



Figure 3-4 Operation with/without MPI

4. Select turning or milling



Figure 3-5 Select turning/milling

Note

If you want to change your selection later, select the directory "mmc2" and copy "dpturn.exe" (turning) or "dpmill.exe" (milling) into the directory "dp.exe".

5. Select drive

Only if several local disk drives are available.

Select the drive for the tmp directory (see screenshot)

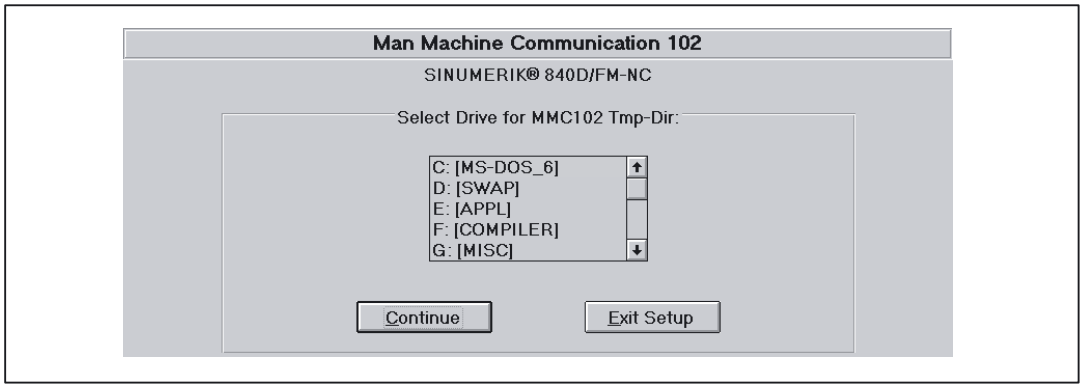


Figure 3-6 Select drive

Otherwise drive C:\ is selected as a default.

Note

The contents of the directory "tmp" are deleted on the installation drive with each restart of MMC 102.

Following the selection, a status display with the inputs made is shown.

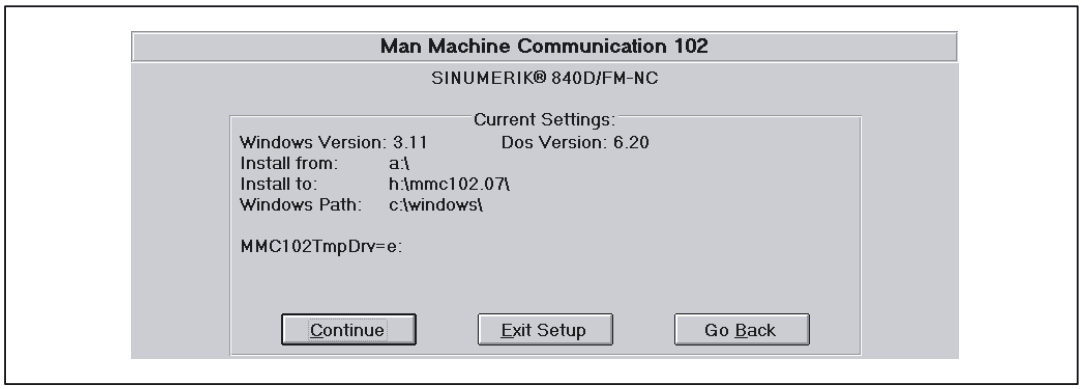


Figure 3-7 Status display of the installation mode

6. Continue

When you press **Continue**, you are prompted to insert the installation diskettes.

Note

Please observe the requests made on the screen.

The program group "SINUMERIK 840D MMC V3.2" is generated.

If the installation is successful, this message appears:

"MMC 102 Installation is complete"

If you want to change the installation path, press **Go back**.

7. Make settings

7a:

OPI interface (1.5 Mbaud), Configuration: **1 MMC** to **1 NCU** (on delivery)

Additional settings are not required.

7b:

MPI interface (187.5 Kbaud), Configuration: **1 MMC** to **1 NCU** (on delivery)

1. Determination of the NCK/PLC bus address
   - If PLC < software version 3.2,

     then NC address = 13

     PLC address = 2
   - If PLC ≥ software version 3.2 and PLC module 314,

     then NC address = 13

     PLC address = 2
   - If PLC ≥ software version 3.2 and PLC module 315,

     then NC address = 3

     PLC address = 2
2. Entering the addresses in files
   - File "S7CFGPGX.DAT"

     In file "S7CFGPGX.DAT" on the MPI driver directory (<installation path>\MMC2\DRV.ID), the following entries must be adapted to the existing hardware configuration by means of an ASCII editor:

     # Setting the interrupt

     "hwint_vector": Setting the interrupt for the MPI card. This interrupt may not be used by another card (e.g. network adapter).

     Default setting: 10.

     # Settings for baud rate

     "baud rate", "tslot" and "tgap": Settings for baud rate. These 3 settings must always be activated/deactivated together by removing/inserting the leading "#" (comment).

     When the baud rate is changed, the setting "ADDRESS1=\PLC, 10000d01" for 1.5 Mbaud or "ADDRESS1=\PLC, 10000201" for 187.5 Kbaud must also be adapted in section [840D] in file <installation path>\MMC2\MMC.ini.

     Default: 1.5 Mbaud.
   - "netnames.ini" file

     The following lines in the file must be changed:

     # bus = opi must be replaced by = **mpi**

     # nck_address = 13 must be replaced by = **3** (if PLC ≥ software version 3.2)

     = 13 (if PLC < software version 3.2)

     # plc_address = 13 must be replaced by **2**

Parallel STEP7/AS300 application

Installation in parallel with the STEP7/AS300 SW can give rise to problems. It may be necessary to reconfigure the drives and restart the system.

3.2.3 Supplementary software conditions

- **Function keys**

  The function keys must not be actuated in any of the displays until the display has fully built up.

- **Monochrome screen**

  When a monochrome screen is used, the colors used by the MMC must be adapted accordingly. For this purpose, select the color scheme "Monochrome" or "Mono positive" in display "Commissioning\MMC\Color setting".

- **User-friendly parameterization**

  The display "Commissioning\MMC\OPI parameters" can now be called up even if there is no link to the NC kernel. This means that the OPI parameters for baud rate and network address can be set easily.

3.2.4 Start program

Program call

The MMC 102/103 software is started on a PG/PC either

- from the program manager through selection of the "SINUMERIK 840D MMC V2.3" program group followed by a **double click** on the **"MMC Startup"** symbol or

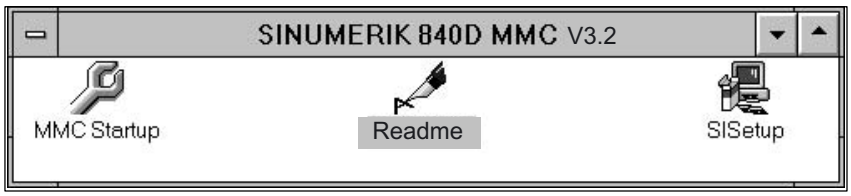


Figure 3-8 SINUMERIK 840D MMC program group

- from the file manager by a **double click** on file **REG_CMD.EXE**.

Communication

If no communication link can be set up to the NCK or 611D, then the message "No communication to NCK" is displayed. If communication is interrupted, e.g. by an NCK reset, then the MMC 102/103 software tries to re-establish the communication link itself.

3.2.5 Close program

Deselect the program

The following steps must be taken to deselect the MMC 102/103 software:

1. Press function key **F10**

   A horizontal softkey bar is displayed.
2. Press function key **Shift + F9**.
3. You can close the program by pressing the **Exit** softkey.

3.3 Operation via PG/PC

3.3.1 General operation

Operating philosophy

The special function keys of the operator keyboard can be used with the full keyboard. Operator inputs can be made using the mouse or via the keyboard.

Keyboard operation

The following table shows the assignments between the function keys and the softkeys/special keys:

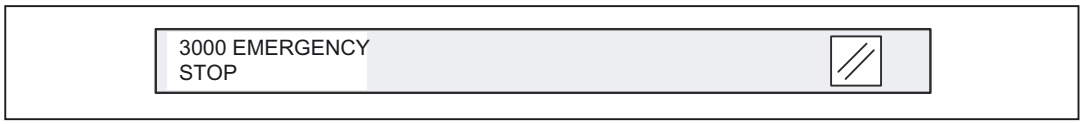
Note

The editor displays only the characters which can be input via the operator panel front keyboard.

Full keyboard	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
with SHIFT	vertic. soft. 1	vertic. soft. 2	vertic. soft. 3	vertic. soft. 4	vertic. soft. 5	vertic. soft. 6	vertic. soft. 7	vertic. soft. 8				
without SHIFT	horiz. soft. 1	horiz. soft. 2	horiz. soft. 3	horiz. soft. 4	horiz. soft. 5	horiz. soft. 6	horiz. soft. 7	horiz. soft. 8				
Full keyboard	5	Esc	Insert	Home	Page Up	Page Down	Enter					
without SHIFT	:	X		?	'	%						

Figure 3-9 Key assignments between operator keyboard and full keyboard

Alarm or message line



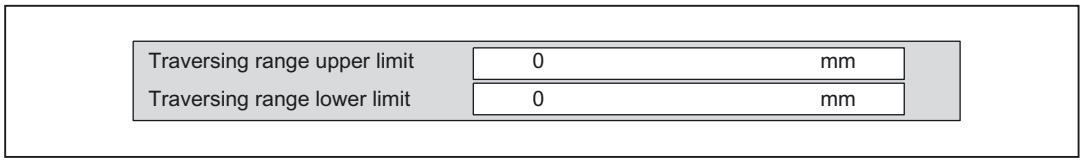
Alarm or message line for displaying information for the operator

i-R

Selection fields **i** and **R**, which appear in every display, have the following meaning:

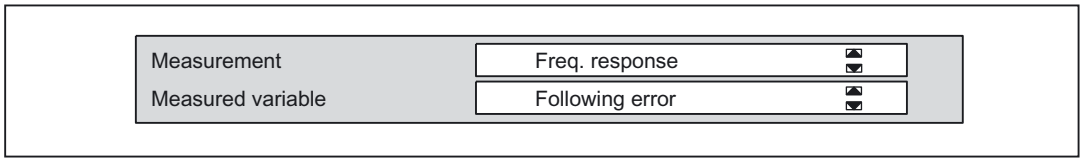
- The **i** field is selected with the **Help** key or by **mouse click**.
- The **R** field is selected with the **F9** key or by **mouse click**. Selection of this field activates the Recall function, i.e. returns the user to the previous level.

Input fields



To allow the input of data, the input cursor is positioned in the appropriate input field by means of the **TAB** or **SHIFT + TAB** key or by **mouse click**. The editing mode is always preset to **Overwrite**. It is possible to switch back and forth between overwrite mode and insert mode using the **Insert** key.

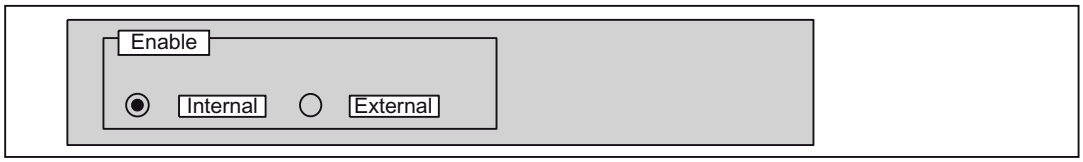
List fields



The functions offered are selected with the cursor keys **UP (")** and **DOWN (#)** or by **mouse click**. The displayed function is valid.

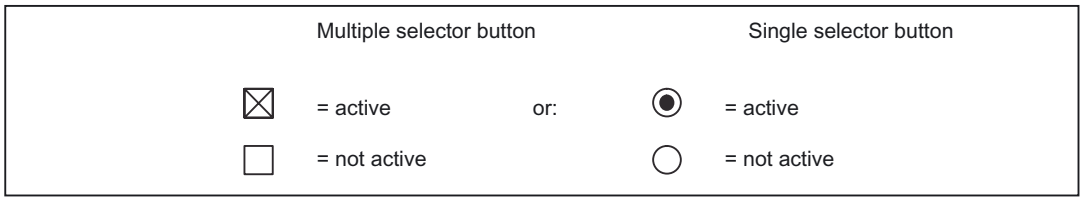
The list fields are selected by means of the **TAB** or **SHIFT + TAB** keys or by **mouse click**.

Single/multiple selector button



The required function is activated with the cursor keys **LEFT (z)** and **RIGHT (!)** or by **mouse click**.

The function fields are selected by means of the **TAB** or **SHIFT + TAB** keys or by **mouse click**.



Activation of fields

To be able to alter values and functions, the window with the input field must be activated using the **CTRL + TAB** keys or the **HOME** key (yellow frame = focus).

3.3.2 Additional information

Axis selection

The "Select axis/Select next axis" inputs in axis-specific displays are always made via the uniformly positioned vertical softkeys **AXIS+** or **AXIS-**.

Function selection/deselection

All functions are activated by means of the **START** softkey and deactivated by means of the **STOP** softkey.

Password

When the **Set password** softkey is selected, a dialog box prompting the user to enter a password appears. Passwords are input as described in:

References:

/FB1/ Function Manual, Basic Functions; Various Interface Signals (A2)

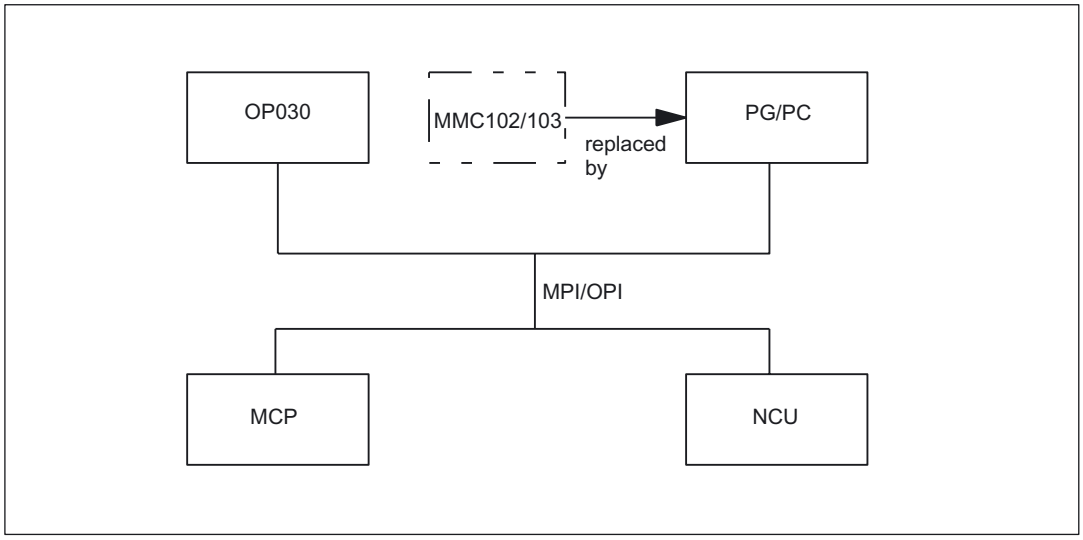
Keyboard assignments

With the exception of keys **F1** to **F12** and **SHIFT + F1** to **F10**, the conditions and key assignments are the same as under WINDOWS™ 3.1.

The key combination **ALT + TAB** can be used at any time to switch from PG/PC operation to other WINDOWS™ applications.

3.3.3 Operation of operator panel fronts

The system responds as follows, for example, when two operator panel fronts are operated in the configuration illustrated below:



1. For the NCU, there is no difference whether the input is from the MMC or OP030 operator panel front.
2. The operator panels are mutually independent in terms of data display, i.e. the display selected on one panel is not affected by the display on the other.
3. Spontaneous events, such as alarms, are displayed on both control units.
4. The protection level with the highest authorization in accordance with the lowest activated protection level number applies to both operator panel fronts.
5. The system does not provide for any further co-ordination between the operator panels.

For further information, please refer to

References:

/FB2/ Function Manual, Extended Functions; Several Operator Panel Fronts and NCUs (B3)

3.4 Simulation of part programs

Windows 32s, version 1.30.166.0 or higher, must be installed in order to operate part program simulation.

For notes on operation, see

References:

/BA/ Operating Manual

3.5 Marginal conditions

The "Operation via PG/PC" function is available in the basic version with software version 3.1 and higher. With software version 3.1, the number of NCUs which may be connected is limited to one and the number of operator panel fronts to two. One of these must be an OP030.

With software version 3.2 and higher, an operator panel front with MMC 100 or MMC 102/103 can also be connected with up to three NCUs.

3.6 Data lists

No signals or machine data are required for this function.

H1: Manual travel and handwheel travel

4

4.1 Brief description

4.1.1 Overview

Applications

Even on modern, numerically controlled machine tools, a facility must be provided that allows the user to traverse the axes manually.

Setting up the machine

This is especially necessary when a new machining program is being set up and the machine axes have to be moved with the traversing keys on the machine control panel or with the electronic handwheel. Where coordinate offset or rotation is selected, manual travel can even be performed in the transformed workpiece coordinate system.

Retraction of tool

After a program interruption caused, for example, by NC STOP, RESET or a power failure, the machine user must retract the tool manually from its current machining position. This is usually done by operating the traversing keys in JOG mode. The transformations and coordinate systems used for machining must remain active while this is done.

Variants of manual travel

The following variants of the manual travel are explained under function description:

- Continuous travel in jog or continuous mode in JOG
- Incremental travel (INC) in jog or continuous mode in JOG
- Travel of axes using electronic handwheels (accessory) in JOG
- Handwheel override in AUTOMATIC (path default and velocity override)

DRF

The differential resolver function (DRF) generates an additional incremental work offset in AUTOMATIC mode via the electronic handwheel. This function can be used, for example, to correct tool wear within a programmed block.

Fixed point approach

The "fixed point approach in JOG" function enables the manual travel to fixed axis positions that are defined through a machine data.

4.1.2 General characteristics of manual travel in JOG

The following is a description of the characteristics which generally apply to manual travel in JOG mode (irrespective of the type selected).

JOG mode

JOG mode must be active if the axes are to be traversed manually (hereafter referred to as "manual travel").

In each case, the active mode is sent to the PLC via interface signal
DB11 DBX4.2 (active mode strobe: JOG).

References:
/FB1/ Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

Machine functions

There are several JOG variants (so-called "machine functions") within JOG mode:

- Continuous (JOG CONT)
- Incremental (JOG INC)
- Jogging with the handwheel.

Handwheel travel

Handwheel travel is also active with the following functions:

- JOG REPOS mode for traversing the geometry and machine axes
- AUTOMATIC mode for moving out a DRF offset
- With path override
- When moving the backlash point of an oscillation

The active machine function is selected via the PLC interface. A separate PLC interface exists for both the machine axes (axis-specific) and the geometry axes (channel-specific).

Simultaneous travel

All axes can be traversed simultaneously in JOG. If several axes are moved simultaneously, there is no interpolatory relation.

Velocity

The velocity for a JOG traversing movement is determined by the following value settings depending on the feedrate mode:

- With active linear feedrate (G94)

  (SD41100 $SN_JOG_REV_IS_ACTIVE (JOG: revolutional/linear feedrate) = 0):

  – With general setting data
  SD41110 $SN_JOG_SET_VELO (axis velocity in JOG)
  or, for rotary axes, with general setting data
  SD41130 $SN_JOG_ROT_AX_SET_VELO
  (JOG velocity for rotary axes)

  or

  – With axial machine data
  MD32020 $MA_JOG_VELO (JOG axis velocity),
  only if SD41110 = 0.

- With active revolutional feedrate (G95)
  (SD41100 $SN_JOG_REV_IS_ACTIVE = 1):

  – With general setting data
  SD41120 $SN_JOG_REV_SET_VELO
  (revolutional feedrate of axes in JOG)

  or

  – With axial machine data
  MD32050 $MD_JOG_REV_VELO (revolutional feedrate in JOG),
  only if SD41120 = 0.

The default setting for feedrate velocity is mm/min or rpm for revolutional feedrate or rotary axes.

Rapid traverse override

If the rapid-traverse-override key is pressed at the same time as the traversing keys, the movement is executed at the rapid-traverse velocity set in the axis-specific machine data (or MD32040 $MA_JOG_REV_VELO_RAPID for revolutional feedrate):

MD32010 $MA_JOG_VELO_RAPID (rapid traverse in jog mode)

Feedrate override

The JOG velocity can also be influenced by the axial feedrate override switch provided that interface signal
DB31, ... DBX1.7 (axial feedrate override active) is active.

Percentages are assigned to the individual feedrate-override switch positions via machine data. For a switch position of 0% the axis is not traversed, provided that 0 is entered in the associated machine data.

Interface signal
DB31, ... DBX1.7 (axial feedrate override active)
has no meaning at switch position 0%.

Instead of being set by the feedrate-override-switch position (gray code), the percentage value (0% to 200%) can be set directly by the PLC. Again, the selection is made via machine data.

References:
/FB1/ Function Manual Basic Functions; Feedrates (V1)

Acceleration

With manual travel, acceleration takes place according to a programmed characteristic. The acceleration characteristic active in JOG for each axis is defined, using the following axial machine data (default setting of axial jerk limitation).

MD32420 $MA_ JOG_AND_POS_JERK_ENABLE

References:
/FB1/ Function Manual, Basic Functions; Acceleration (B2)

Display

The JOG main screen appears when JOG mode is selected. This main screen contains values relating to position, feedrate, spindle, and tool.

For clarifications on the individual display values, please refer to:
References:
/BA/ SINUMERIK 840D HMI User Manual

Coordinate systems

In JOG mode, the user has the option to traverse axes in different coordinate systems.

The following coordinate systems are available:

- Basic coordinate system

  Each axis can be traversed manually.

- Workpiece coordinate system

  Only geometry axes can be traversed manually (channel-specific).

Geometry axes

In manual travel, a distinction must be made as to whether the affected axis is to be traversed as a machine axis (axis-specific) or as a geometry axis (channel-specific).

First we will focus on the characteristics of machine axes. Special features relating to manual traversal of geometry axes are described in more detail in "Geometry-axis manual travel".

Spindle manual travel

Spindles can also be traversed manually in JOG mode. Essentially, the same conditions apply as for manual travel of axes. Spindles can be traversed in JOG mode using the traversing keys continuously or incrementally, in jog or continuous mode, or using the handwheel. The mode is selected and activated via the axis-/spindle-specific PLC interface as for the axes. The axis-specific machine data also apply to the spindles. Special features relating to manual traversal of spindles are described in more detail in "Special features of spindle manual travel".

4.1.3 Control of manual-travel functions via PLC interface

HMI/NCK/PLC interface

Most individual functions required for manual travel in JOG are activated via the PLC user interface. The machine manufacturer can adapt the manual-travel functionality to the machine tool depending on the configuration, using the PLC user program.

Machine control panel

The signals between the machine control panel and the individual PLC/NCK-interface data blocks can be transferred by the PLC user program on a machine-specific basis. The PLC user program defines the assignment of the direction keys on the machine control panel to the axis/spindle (machine axes, geometry axes) traversing keys.

The following machine-control-panel signals are of particular importance to manual travel:

- JOG mode (selection)
- Machine function INC1 ...
- Direction keys
- Feedrate override and spindle-speed override

For further information about machine-control-panel signal transmission, see:

References:
/FB1/ Function Manual Basic Functions; PLC Basic Program (P3)

Selection of machine function

The machine functions available in JOG mode can be selected from the following locations:

Via machine control panel (MCP)	→ e.g., user DB interface
Via PLC user program	→ PLC/NCK interface

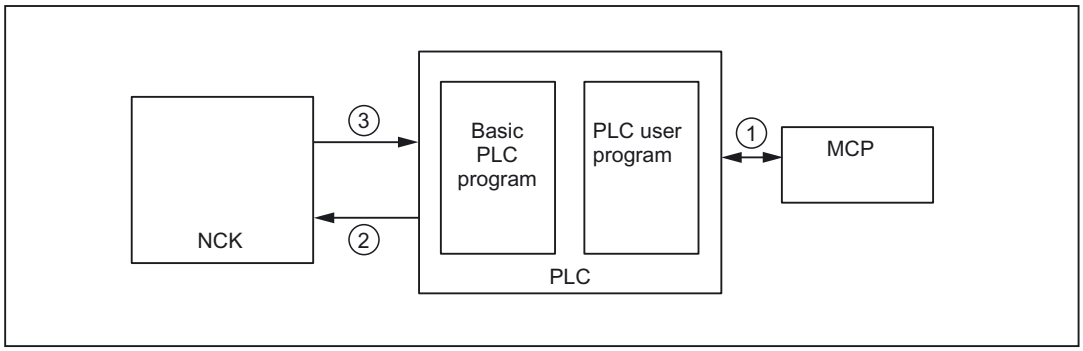
The PLC user program transfers the machine function pending at the machine-control-panel interface to the relevant PLC/NCK interface. Here the axis-specific NCK/PLC interface should be used for a machine axis/spindle, and the channel-specific NCK/PLC interface should be used for a geometry axis:

4.1.4 Control-system response to power ON, mode change, RESET, block search, REPOS

A RESET will always abort (with braking ramp) any traversing movement triggered by handwheel travel.

Selection from MCP

The following example shows the sequence of operations for selecting the "continuous" machine function for a machine axis of the machine control panel.



① The user selects the "Continuous JOG" machine function on the machine control panel for a machine axis.

② "Machine function" interface signal

The PLC program (basic or user program) logically combines this IS and sends the following request to the NCK interface
DB31, ... DBX5.6 (continuous machine function).

Before this happens, the PLC user program first checks that this request is permissible with regard to the current machine status.

③ "Active machine function" interface signal

The machine function is selected within the control.

As soon as the "continuous JOG" (DB31, ... DBX65.6) machine function is active, this is signaled to the PLC by the NCK.

Figure 4-1 Sequence of operations for selecting machine functions of the machine control panel

For more information about signal transmission between the MCP and PLC, see:
References:
/FB1/ Function Manual Basic Functions; Basic PLC Program (P3)

4.2 Continuous travel

4.2.1 General functionality

Selection

In JOG mode, continuous travel must be activated via the PLC interface:
DB21, ... DBX13.6, ff (machine function: continuous)

As soon as continuous travel is active, interface signal
DB21, … DBX41.6, ff (active machine function: continuous)
is returned to the PLC. continuous)
returned.

Traversing keys +/-

The plus and minus traversing keys are selected to move the relevant axis in the appropriate direction. If both traversing keys are pressed simultaneously, there is no traversing movement, or, if an axis is in motion, it is stopped.

Note

When the control is switched on, axes can be traversed to the limits of the machine because they have not yet been referenced. This can cause emergency limit switches to be triggered.

The software limit switches and the working-area limitation are not operative.

Motion command +/-

As soon as a travel request is pending for an axis (e.g., once a traversing key has been pressed), the
DB21, ... DBX40.7 (motion command +)
or
DB21, … DBX40.6 (motion command -)
interface signal is output, depending on the direction of movement.

4.2.2 Distinction between inching mode continuous mode

Selection

In JOG mode, we distinguish between traversing in inching mode and in continuous mode.

The selection is made using general setting data
SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD
(inching mode/continuous mode in continuous JOG)
and applies to all axes.

Default setting

Traversing in inching mode is the default setting.

Continuous travel in jog mode

Function

In inching mode (default setting), the axis traverses for as long as the traversing key is held down, provided that no axis limitation is reached. When the traversing key is released, the axis is decelerated to zero speed and the movement comes to an end.

Continuous traversing in continuous mode

Function

When the traversing key is pressed and quickly released (first rising edge), the axis starts to traverse at the set velocity in the desired direction. This traversing movement is continued even after the traversing key is released. The movement of the axis is stopped either by the user or because of a response within the control (e.g., software limit switch reached).

⚠ WARNING
If "continuous" mode is selected, several axes can by started by pressing and releasing the relevant direction key. Any interlocks must be implemented via the PLC!

Interrupt traversing movement

The user can use the following methods to interrupt the traversing movement:

- Setting feedrate override to 0%
- Axial feed disable (PLC interface signal)
- NC STOP or NC STOP axis/spindle

If the cause of the interruption is removed, the axis continues to traverse.

Abort traversing movement

The traversing movement can be stopped and aborted by means of the following operations or monitoring functions:

- Pressing the same traversing key again (second rising edge)
- Pressing the traversing key for the opposite direction
- RESET

- On deselection of the continuous mode
- On reaching the first valid limitation
- In the event of faults

⚠ CAUTION
Software limit switches and working-area limitations are only activated after reference point approach.

Note

While an axis is moving, a change of mode from JOG to AUT or MDI is not permitted within the control.

4.2.3 Special features of continuous travel

Indexing axes

When an axis that is declared as an indexing axis is traversed in continuous mode, it always traverses to indexing positions. For example, the axis traverses on to the next indexing position in the direction of travel even if the traversing key is released in inching mode.

References:
/FB2/ Function Manual, Extended Functions; Indexing Axes (T1)

4.3 Incremental travel (INC)

4.3.1 General functionality

Programming increments

The path to be traversed by the axis is defined by so-called increments (also called "incremental dimensions"). The required increment must be set by the machine user before the axis is traversed.

The setting is made on the machine control panel, for example. Once the corresponding logic operation has been performed, the
DB31, ... DBB5 ff (machine function: INC1 to INCvar)
interface signal associated with the required increment should be set by the PLC user program.

Settable increments

The user can set up to six different increment sizes:

- Five fixed increments

  The increment size is defined collectively for all axes with the following general machine data:

  MD11330 $MN_JOG_INCR_SIZE_TAB (increment size INC/handwheel)

  INC1, INC10, INC100, INC1000, and INC10000 are the default settings.

- One variable increment (INCvar)

  The increment setting for the variable increment can also be made for all axes using general setting data (size of the variable increment for INC/handwheel).

Increment weighting

The path weighting of **one** JOG increment is defined using axial machine data
MD31090 $MA_JOG_INCR_WEIGHT
(weighting of an increment of a machine axis for INC/handwheel).

4.3.2 Distinction between inching mode and continuous mode

Selection

When machine axes are in incremental mode, we also distinguish between inching mode and continuous mode.

The selection is made using general machine data
MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in inching mode).

Inching mode is the default setting.

Incremental travel in inching mode

Function

If the traversing key for the required direction (e.g., +) is pressed, the axis begins to traverse the increment that has been set. If the traversing key is released before the increment has been fully traversed, the movement is interrupted and the axis stops. If the same traversing key is pressed again, the axis traverses the remaining distance until it is zero. Up to this point, the movement can still be interrupted by releasing the traversing key.

Pressing the traversing key for the opposite direction does not have any effect, unless the increment has been fully traversed or the movement has been aborted.

Abort traversing movement

If you do not want to traverse the whole increment, the traverse movement can be aborted with RESET or the interface signal
DB31, ... DBX2.2 (delete distance-to-go).

Incremental travel in continuous mode

Function

The axis traverses the entire set increment when the traversing key is pressed (first rising edge). If the same traversing key is pressed again (second rising edge) before the axis has finished traversing the increment, the traversing movement is aborted, i.e., not completed.

Interrupt traversing movement

Behavior as for continuous travel.

Abort traversing movement

The traversing movement can be stopped and aborted by means of the following operations or monitoring functions:

- Pressing the same traversing key again (second rising edge)
- Pressing the traversing key for the opposite direction
- RESET
- Deleting axial distance-to-go (PLC interface signal)
- On reaching the first valid limitation

⚠ CAUTION

Software limit switches and working-area limitations are only activated after reference point approach.

- On deselection or change of the current increment (e.g., change from INC100 to INC10)
- In the event of faults (e.g., on cancellation of the servo enable)

Note

While an axis is moving, a change of mode from JOG to AUT or MDI is not permitted within the control.

⚠ WARNING

If "continuous" mode is selected, several axes can by started by pressing and releasing the relevant direction key. Any interlocks must be implemented via the PLC!

4.3.3 Special features of incremental travel

Indexing axes

Irrespective of the current set increment value, an axis that is declared as an indexing axis (MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB (axis is indexing axis)) traverses to the next higher indexing position when the "+" traversing key is pressed.
Similarly, pressing the "-" traversing key causes the next lower indexing position to be approached.

References:
/FB2/ Function Manual, Extended Functions; Indexing Axes (T1)

4.4 Handwheel travel in JOG

4.4.1 General functionality

Selection

JOG mode must be active. The user must also set the increment INC1, INC10, etc., which applies to handwheel travel. As with incremental travel, the required machine function must be set at the PLC interface accordingly.

Traversing

When the electronic handwheel is turned, the associated machine axis is traversed either in the positive or negative direction depending on the direction of rotation.

Traversing path

The traverse path produced by rotating the handwheel is dependent on the following factors:

- Number of handwheel pulses received at the interface
- Active increment (machine function INC1, INC10, INC100, ... INCvar)
- Pulse weighting of the handwheel using general machine data
  MD11320 HANDWH_IMP_PER_LATCH (handwheel pulses per detent position)
- Weighting of an increment for INC/handwheel
  (MD31090 $MA_JOG_INCR_WEIGHT)

Motion command +/-

During the axis movement, the
DB21, ... DBX40.7 (motion command +
or
DB21, … DBX40.6 (motion command -)
interface signal is output to the PLC, depending on the direction of movement.

If the axis is already being moved using the traversing keys, the handwheel cannot be used.

Handwheel connection

Two handwheels can be connected simultaneously. In this way, up to two axes can be traversed by handwheel simultaneously.

Exception: If several axes are assigned to one handwheel simultaneously, more than two axes can be traversed with handwheels.

Handwheel assignment

A handwheel is assigned to a geometry or machine axes via a separate axis-specific VDI interface signal.

The axis to be moved as a result of rotating handwheel 1 or 2 is set as follows:

- Via the PLC user interface using the
  DB31, ... DBX4.0 - DBX4.2 (activate handwheel) interface signal.

  (for geometry axis: DB21, ... DBX12.0 - 12.2 ff)

or

- using menu-assisted operation (HMI)

  Pressing the "Handwheel" softkey in the JOG-mode basic menu displays the "Handwheel" window. Here, every handwheel can be assigned an axis and the handwheel can be enabled or disabled.

The assignment is linked to the PLC interface through the PLC user program. In this way, several axes can be assigned to one handwheel simultaneously.

Function

The electronic handwheels (accessories) can be used to simultaneously traverse selected axes manually. The weighting of the handwheel graduations is dependent on the increment-size weighting. Where coordinate offset or rotation is selected, manual traversal can even be performed in the transformed workpiece coordinate system.

Handwheel selection by HMI

A separate user interface is provided between the HMI and PLC to allow activation of the handwheel from the operator-panel front. This interface supplied by the basic PLC program for handwheels 1 and 2 contains the following information:

- Axis numbers assigned to the handwheel:
  DB10 DBB100 ff (axis number of handwheel n)
- Additional information about the machine or geometry axis:
  DB10 DBX100.7 ff (machine axis)
- The channel number assigned to the handwheel, if a geometry axis was selected during handwheel selection:
  DB10 DBX97 ff (channel number of geometry axis of handwheel n)
- Information about whether the handwheel is enabled or disabled:
  DB10 DBX100.6 ff (handwheel deselected)

For the specified axis, the associated interface signal:
DB31, ... DBX4.0 - DBX4.2 (activate handwheel)
(for geometry axis: DB21, ... DBX12.0 - 12.2 ff)
is either set to "0" (lock) or to "1" (enable) from the basic PLC program.

Input frequency

The handwheel connections can receive handwheel pulses with a maximum input frequency of 100 kHz.

Velocity

In handwheel travel the following axis velocities, effective during JOG mode, are used:

- SD41110 $SN_JOG_SET_VELO (axis velocity for JOG)
- SD41130 $SN_JOG_ROT_AX_SET_VELO (axis velocity for rotary axes for JOG mode)
- MD32020 $MA_JOG_VELO (conventional axis velocity)

Because of the limited feedrate, the axis is not able to follow the handwheel rotation synchronously, especially in the case of a large pulse weighting, and therefore overtravels.

Abortion of traversing movement

The traversing movement is aborted by performing a RESET or using the
DB31, ... DBX2.2 (delete distance-to-go/spindle reset)
interface signal.
The setpoint/actual-value difference is deleted.

The traversing movement is only interrupted with STOP.
Any setpoint/actual-value difference is retained.

The distance-to-go is then traversed using START.

Increment-value limitation

The user can limit the size of the increment selected for geometry axes via channel-specific machine data
MD20620 $MC_HANDWH_GEOAX_MAX_INCR_SIZE
(limitation of handwheel increment for geometry axes).

The size of the selected increments for machine axes can be limited using the following axial machine data:

MD32080 $MA_HANDWH_MAX_INCR_SIZE (limiting the selected increment)

A traversing movement defined by the handwheel for a **geometry axis** is defined by:

- Traversing path
- Size of the variable increment
  (SD41010 $SN_JOG_VAR_INCR_SIZE)
- Geometry-axis assignment
  (MD20620 $MC_HANDWH_GEOAX_MAX_INCR_SIZE)

A traversing movement defined by the handwheel for a **machine axis** is defined by:

- Traversing path
- Size of the variable increment
  (SD41010 $SN_JOG_VAR_INCR_SIZE)
- Machine-axis assignment
  (MD32080 $ HANDWH_MAX_INCR_SIZE)

Movement in the opposite direction

Depending on machine data:
MD11310 $MN_HANDWH_REVERSE (threshold for handwheel change of direction),
the response to a change of the traversing direction is as follows:

- If the handwheel is moved in the opposite direction, the resulting distance is computed and the calculated end point is approached as fast as possible.

  If this end point is located before the point where the moving axis can decelerate in the current direction of travel, the unit is decelerated and the end point is approached by moving in the opposite direction. If this is not the case, the newly calculated end point is approached immediately.
- If the handwheel is moved in the opposite direction by at least the number of pulses indicated in the machine data, the axis is decelerated as fast as possible and all pulses received until the end of the interpolation are ignored.

  This means another movement takes place only after the axis reaches zero speed (setpoint side).

Acceleration

With manual travel, acceleration takes place according to a programmed characteristic. The acceleration characteristic effective for JOG for the individual axis is defined, using the following axial machine data:

MD32420 $MA_ JOG_AND_POS_JERK_ENABLE (basic setting of the axial jerk limiting)

Reference:
Function Manual, Basic Functions; Acceleration (B2)

Response at software limit switches, working-area limitation

When axes are traversed in JOG mode, they can traverse only up to the first active limitation before the corresponding alarm is output.

Depending on machine data:
MD11310 $MN_HANDWH_REVERSE (threshold for direction change, handwheel)
the response is then as follows (as long as the axis has still not arrived at the end point from the setpoint side):

- The distance resulting from the handwheel pulses forms a fictitious end point which is used for subsequent calculations.

  If this fictitious end point is positioned, for example, 10 mm behind the limitation, these 10 mm must be traversed in the opposite direction before the axis traverses again. If a movement in the opposite direction is to be performed immediately after a limitation is

reached, the fictitious distance-to-go can be deleted via delete distance-to-go or deselection of the handwheel assignment.

- All handwheel pulses leading to an end point behind the limitation are ignored. Any movement of the handwheel in the opposite direction leads to an immediate movement in the opposite direction, i.e., away from the limitation.

Limitations

The limitations are also active when traversing with the handwheel.

For additional information, refer to the "Monitoring" subject.

Revolutional feedrate

In JOG mode, the response of the axis/spindle also depends on the following setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE (JOG: revolutional/linear feedrate)

SD41100 $SN_JOG_REV_IS_ACTIVE (JOG: revolutional/linear feedrate)	
Active	An axis/spindle is always traversed with revolutional feedrate MD32050 $MA_JOG_REV_VELO (revolutional feedrate for JOG) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate for JOG with rapid traverse override), depending on the master spindle.
Not active	The response of the axis/spindle depends on setting data SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles).
	The response of a geometry axis that acts on a frame with rotation depends on channel-specific setting data SD42600 $SC_JOG_FEED_PER_REV_SOURCE (control of revolutional feedrate in JOG). (In JOG mode, revolutional feedrate for geometry axes on which a frame with rotation is effective.)

4.4.2 Travel request

Compared to the previous response, additional options are possible with the travel-request signals, as described in the following.

"Travel-request" signals

DB21, … DBX40.5	Travel request +	Geometry axis 1
DB21, … DBX40.4	Travel request -	Geometry axis 1
DB21, … DBX46.5	Travel request +	Geometry axis 2
DB21, … DBX46.4	Travel request -	Geometry axis 2
DB21, … DBX52.5	Travel request +	Geometry axis 3
DB21, … DBX52.4	Travel request -	Geometry axis 3
DB21, … DBX332.5	Travel request +	Orientation axis 1
DB21, … DBX332.4	Travel request -	Orientation axis 1
DB21, … DBX336.5	Travel request +	Orientation axis 2
DB21, … DBX336.4	Travel request -	Orientation axis 2
DB21, … DBX340.5	Travel request +	Orientation axis 3
DB21, … DBX340.4	Travel request -	Orientation axis 3
DB31, … DBX64.5	Travel request +	From axis/spindle
DB31, … DBX64.4	Travel request -	From axis/spindle

Handwheel travel with path default

If a pending stop condition is **not an abort criterion**
(see
MD32084 $MA_HANDWH_STOP_COND
or
MD20624 $MC_HANDWH_CHAN_STOP_COND)
during handwheel travel with path default
(MD11346 $MN_HANDWH_TRUE_DISTANCE == 1 or == 3),
the "travel request" and "motion command" PLC signals are output in accordance with the general response (see figures below).

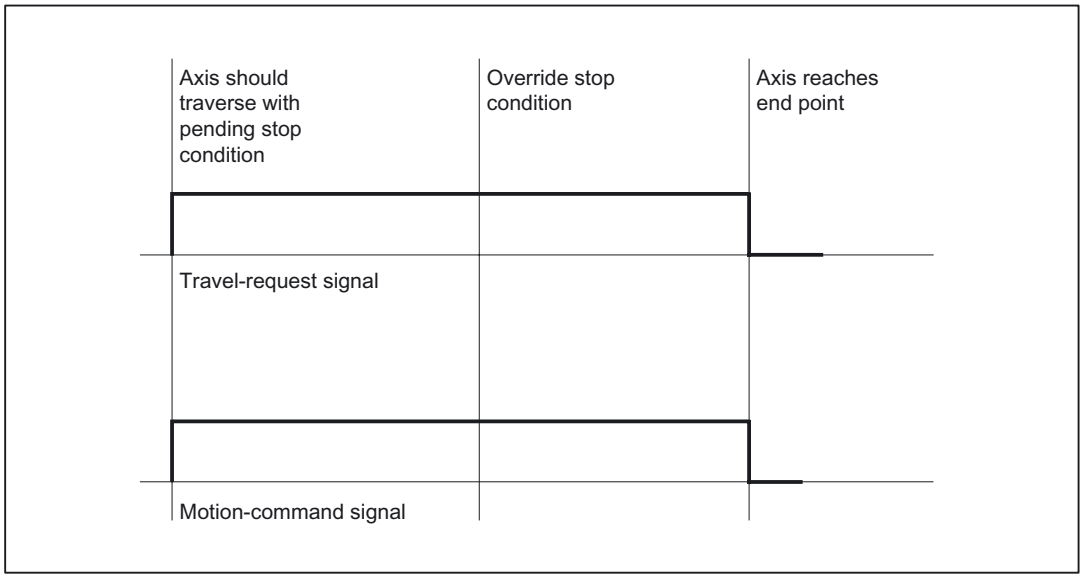


Figure 4-2 Signal/timing diagram MD17900 $MN_VDI_FUNCTION_MASK bit 0 = 0

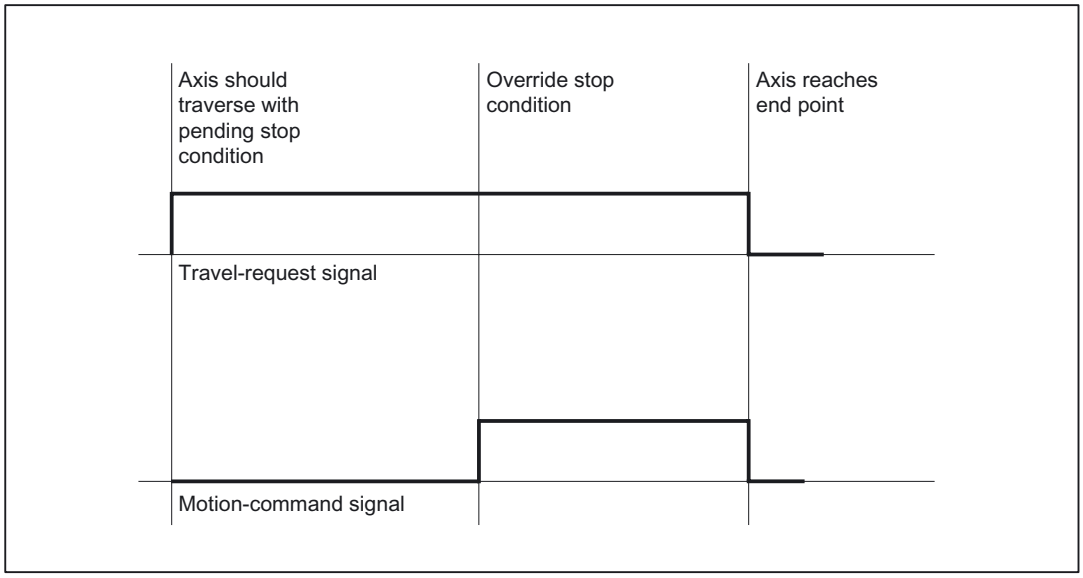


Figure 4-3 Signal/timing diagram MD17900 $MN_VDI_FUNCTION_MASK bit 0 = 1

If a pending stop condition is selected as an **abort criterion** via machine data
MD32084 $MA_HANDWH_STOP_COND
or
MD20624 $MC_HANDWH_CHAN_STOP_COND
during handwheel travel, once again **no motion command** is output (compatibility), **but** the corresponding **travel request** is output.

When the stop condition is overridden, the corresponding "travel request" PLC signal is reset, as an abort is present. The stop condition is no longer active, but the axis cannot be traversed as the stop condition has caused an abort.

In addition, either the path default
(MD11346 $MN_HANDWH_TRUE_DISTANCE == 1 or == 3)
is active, or the handwheel is moved continuously, i.e., provides pulses.

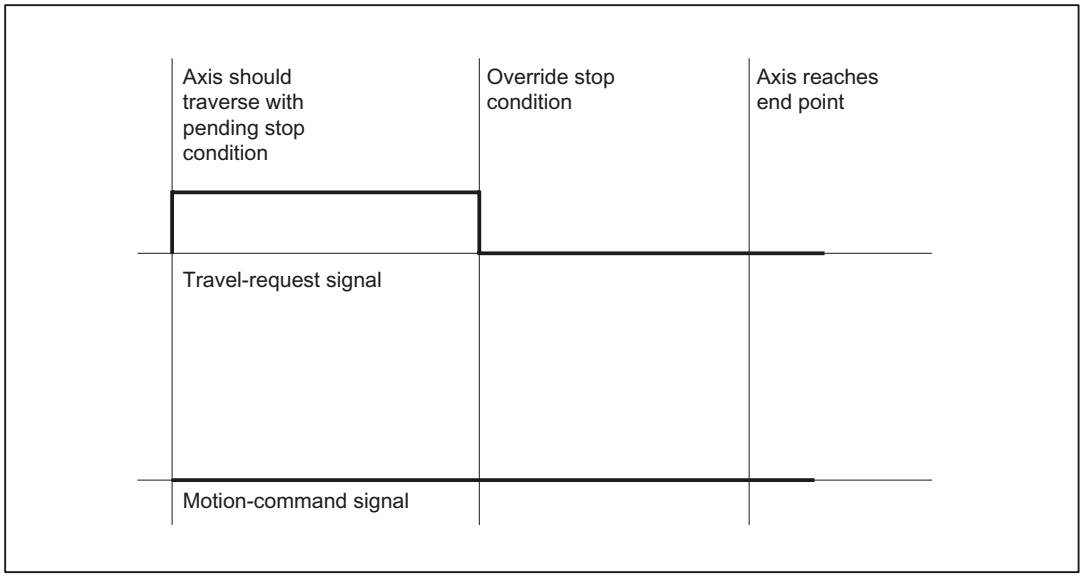


Figure 4-4 Signal/timing diagram, handwheel travel when stop condition is abort criterion

If a stop condition is activated during the handwheel-travel movement, the movement is aborted and the "travel request" and "motion command" are reset.

With velocity specification

If the handwheel is no longer moved with velocity specification (MD11346 $MN_HANDWH_TRUE_DISTANCE == 0 or == 2), the "travel request" PLC signal is reset.

The "travel request" PLC signal is also reset when the handwheel is deselected.

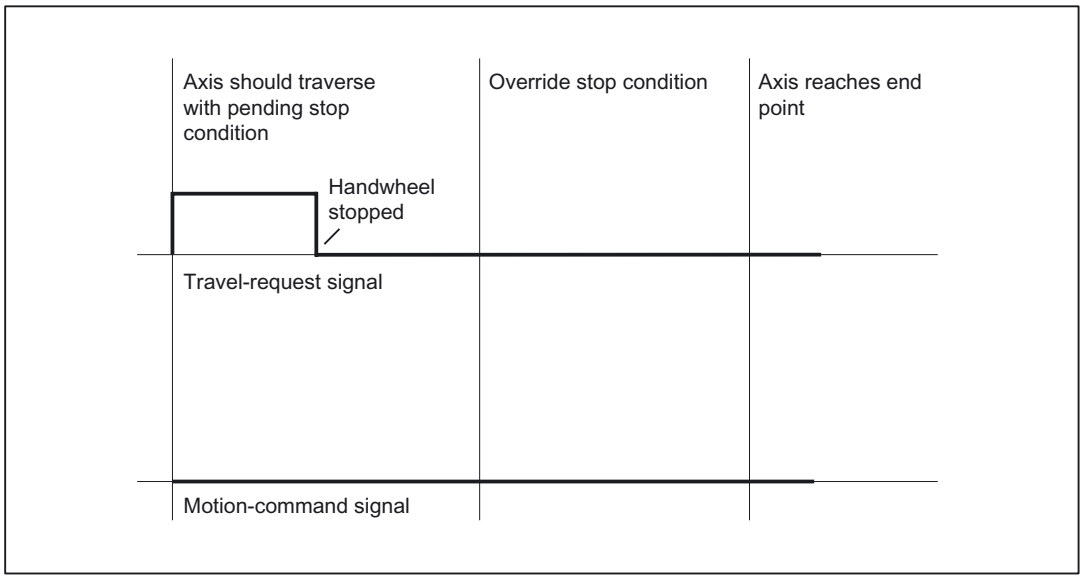


Figure 4-5 Signal/timing diagram, handwheel travel with velocity specification when stop condition is abort criterion

Supplementary conditions

With NC Stop present, no motion command and, therefore, no travel request is output. There is an exception with DRF travel:

If DRF travel is permitted in the NC-Stop state via machine data
MD20624 $MC_HANDWH_CHAN_STOP_COND (bit 13 == 1),
the response corresponds to that of handwheel travel.

As for the motion command, the travel request is the sum of all the sub-movements, i.e., the component from couplings and offset values is also taken into account.

Examples

In machine data
MD32084 $MA_HANDWH_STOP_COND (control of VDI signals relating to handwheel)
the feed stop is set as the abort criterion.
The "feed stop" PLC signal is present. Handwheel travel is selected (JOG mode, DRF travel in AUTOMATIC mode).

The handwheel is turned in the positive direction:	The "travel request +" PLC signal is output from axis/spindle; no motion command + is output from axis/spindle.
The "feed stop" PLC signal is reset:	No travel request, no motion command

4.4.3 Double use of the handwheel

Alarm 14320

The double use of a handwheel for DRF and velocity or distance overlay, including contour handwheel, is suppressed and is displayed using the self-clearing alarm 14320 (Handwheel %1 used twice (%2) in channel %3 axis %4), if, different influences can act on an axis as a result of the handwheel.

This means that an overlaid movement can only be executed when no DRF offset (triggered by the same handwheel) is active for the axes in the basic coordinate system that are involved in the movement, i.e., the DRF movement must have been terminated.

If an overlaid movement has been started, no DRF offset can be started for any of the axes involved that are supplied by the same handwheel. Such a DRF movement is only possible when the movement with overlay has reached its end point or has been aborted by delete distance-to-go or RESET.

If the handwheel override and DRF offset are to be active simultaneously, this is possible with activation of **two** separate handwheels.

Example: Path override

Assumption:
Channel 1 and geometry axis X correspond to machine axis 3 and geometry axis Y corresponds to machine axis 5 and handwheel 2 is selected for the first geometry axis.

If block `X10 Y10 FD=0` is processed in the main run, neither machine axis 3 nor machine axis 5 can be traversed with DRF via handwheel 2. If handwheel 2 is assigned to machine axis 3 while the channel-specific DRF signal is active, then alarm 14320 (Handwheel 2 used twice (8) in channel 1 axis X) is signaled.

If machine axis 3 or machine axis 5 is traversed with DRF using the 2nd handwheel, then motion `X10 Y10 FD=0` cannot be executed and alarm 14320 (handwheel 2 used twice (3) in channel 1 axis X) or alarm 14320 (handwheel 2 used twice (3) in channel 1 axis Y) is signaled.

Example: Velocity override of positioning axis

Assumption:
Channel 1: Channel axis A corresponds to machine axis 4 and handwheel 1 is assigned to this axis.

If block `POS[A]=100 FDA[A]=0` is processed in the main run, machine axis 4 cannot be traversed with DRF. This means that if the channel-specific DRF signal is active, alarm 14320 (Handwheel 1 used twice (6) in channel 1 axis A) is signaled.

If machine axis 4 is traversed with DRF, then no `POS[A]=100 FDA[A]=0` movement can be executed while a DRF movement is being performed. Alarm 14320 (Handwheel 1 used twice (1) in channel 1 axis A) is signaled.

Example: Distance overlay PLC axis (840D sl)

Assumption:
Channel 1: Handwheel 2 is assigned to machine axis 4.

If an axis movement with path override of the 4th machine axis triggered by FC18 is processed in the main run, machine axis 4 cannot be traversed with DRF. This means that if the channel-specific DRF signal is active, alarm 14320 (Handwheel 2 used twice (9) in channel 1 axis A) is signaled.

If machine axis 4 is traversed with DRF, then no axis movement with path override triggered by FC18 can be executed while a DRF movement is being performed. Alarm 14320 (Handwheel 2 used twice (4) in channel 1 axis A) is signaled.

4.5 Handwheel override in automatic mode

4.5.1 General functionality

Function

With this function it is possible to traverse axes or to change their velocities directly with the handwheel in automatic mode (Automatic, MDI).

The handwheel override is activated in the NC part program using the NC language elements `FD` (for path axes) and `FDA` (for positioning axes) and is **non-modal**.

With positioning axes, it is possible to activate the handwheel override modally using traverse instruction `POSA`. When the programmed target position is reached, the handwheel override becomes inactive again.

Additional axes can be traversed simultaneously or using interpolation in the same NC block.

The concurrent-positioning-axes function can also be activated by the PLC user program.

Distinction

Depending on the programmed feedrate, a distinction is made between the following for handwheel override:

- **Path definition**

  Axis feedrate = 0 (FDA = 0)

- **Velocity override**

  Axis feedrate > 0 (FD or FDA > 0)

The table below shows which axis types can be influenced by the "handwheel override in automatic mode" function.

Axes that can be influenced by the "handwheel override in automatic mode" function		
Axis type	**Velocity override**	**Path definition**
Positioning axis	FDA[AXi] > 0 ; acts axially	FDA[AXi] = 0
Concurrent positioning axis	Parameter "Handwheel override active" = 1 and axis feedrate > 0 from FC18	Parameter "Handwheel override active" = 1 and axis feedrate = 0 from FC18
Path axis	FD > 0 ; acts on path velocity	Not possible

Path definition

With axis feedrate = 0 (e.g., `FDA[AXi] = 0`), the traversing movement of the positioning axis towards the programmed target position is controlled entirely by the user rotating the assigned handwheel.

The direction in which the handwheel is turned determines the traversing direction of the axis. The programmed target position cannot be exceeded during handwheel override. The axis can also be moved toward the programmed target position from the opposite direction, whereby the movement is only restricted by the axial position limitations.

A block change occurs when:

- The axis has reached the programmed target position

or

- The distance-to-go is deleted by axial interface signal DB31, ... DBX2.2 (delete distance-to-go).

From this moment on, the path default is deactivated and any further handwheel pulses have no effect.

After this, incrementally programmed positions refer to the point of interruption and not to the last programmed position.

Velocity override

With regard to the velocity override, a distinction is made between axis feedrate and path feedrate.

- **Axis-velocity override** (FDA[AXi] > 0):

  The positioning axis is moved to the target position at the programmed axial feedrate. Using the assigned handwheel, it is possible to increase the axis velocity or to reduce it to a minimum of zero depending on the direction of rotation. The resulting axis feedrate is limited by the maximum velocity. However, the axis cannot be traversed in the opposite direction to that programmed.

  The block change is performed as soon as the axis reaches the programmed target position. This causes the velocity override to be deactivated automatically and any further handwheel pulses have no effect.

  Similarly, this also applies to concurrent positioning axes, where the target position and the velocity are defined by the PLC.

- **Path-velocity override** (FD > 0):

  The path axes programmed in the NC block traverse to the target position at the programmed feedrate. If the velocity override is active, the programmed path velocity is overridden by the velocity generated with the **handwheel of the first geometry axis**. The block change is performed as soon as the programmed target position is reached.

  The path velocity is increased or reduced to a minimum of zero depending on the direction of rotation of the handwheel. However, it is not possible to reverse the direction of movement with handwheel override.

Application example

The "Handwheel override in automatic mode" function is frequently used on grinding machines. For example, the user can position the reciprocating grinding wheel on the workpiece using the handwheel (path default). After scratching, the traversing movement is terminated and the block change is initiated (by activating DB31, ... DBX2.2 (delete distance-to-go)).

Preconditions

In order to activate "Handwheel override in automatic mode", the following preconditions must have been met:

- A handwheel must be assigned to the axis in question.
- Pulse weighting exists for the assigned handwheel.

Handwheel assignment

The connected handwheels are assigned to the axes via the PLC user interface using the DB31, ... DBX4.0 to DBX4.2 (activate handwheel)
interface signal, in a similar manner as for "handwheel travel in JOG", or using menu-assisted operation.

If handwheel override is programmed for an axis to which no handwheel is assigned, a distinction is made between the following cases:

- **For velocity override:**

  The axes traverse at the programmed velocity.
  A self-acknowledging alarm is output (without response).

- **For path definition:**

  No traversing movement is performed because the velocity is zero.
  A self-acknowledging alarm is output (without response).

Note

When the velocity override is applied to path axes, only the **handwheel of the first geometry axis** acts on the path velocity.

Handwheel weighting

The traverse path of the axis that is generated by rotating the handwheel by one detent position is dependent on several factors (see "Handwheel travel in JOG"):

- Selected increment size:

  MD11330 $MN_JOG_INCR_SIZE_TAB[5] (increment size for INC/handwheel)
  or
  SD41010 $SN_JOG_VAR_INCR_SIZE (size of the variable increment for JOG)

- Weighting of an increment:

  MD31090 $MA_JOG_INCR_WEIGHT

- Number of handwheel pulses per detent position:

  MD11320 $MN_HANDWH_IMP_PER_LATCH

For example, the axis traverses by 0.001 mm per handwheel detent position if machine function INC1 and the default setting of the above machine data are selected.

In the case of velocity override, the velocity results from the traverse path covered using the handwheel within a certain period of time.

Example

Assumptions:
The operates turns the handwheel with 100 pulses/second.

The selected machine function is INC100.

The default setting is made for the above machine data for handwheel weighting.

⇒ Handwheel traverse path per second:	10 mm
⇒ Velocity override:	0.6 m/min

PLC interface signals

As soon as the handwheel override takes effect, the following interface signals to the PLC are set to signal 1:

- **For positioning axes:**

  DB31, ... DBX62.1 (handwheel override active)

- **For concurrent positioning axes:**

  DB31, ... DBX62.1 (handwheel override active)

- **For path axes:**

  DB21, ... DBX33.3 (handwheel override active)

For the path default, depending on the traversing direction, the corresponding DB31, ... DBX64.6 and 64.7 (motion commands +/-) interface signals are output to the PLC.

Limitations

The axial limitations (software limit switch, hardware limit switch, working-area limitation) are effective in conjunction with handwheel override. With path default, the axis can be traversed with the handwheel in the programmed traversing direction only as far as the programmed target position.

The resulting velocity is limited by the axial machine data:

MD32000 $MA_MAX_AX_VELO(maximum axis velocity)

NC Stop/override = 0

If the feedrate override is set to 0% or an NC Stop is initiated while the handwheel override is active, the following applies:

- **For path definition:**

  The handwheel pulses arriving in the meantime are summated and stored. If NC Start or the feedrate override > 0%, the saved handwheel pulses become effective (i.e., are traversed).

  If, however, the handwheel is deactivated prior to this
  (DB21, ... DBB12/16/20 (activate handwheel n)),
  the saved handwheel pulses are deleted.

- **For velocity specification:**

  The handwheel pulses arriving in the meantime are not summated and are not active.

4.5.2 Programming and activating handwheel override

General information

When the handwheel override is programmed with NC language elements `FD` (for path axes) and `FDA` (for positioning axes), the following points must be observed:

- `FDA` and `FD` function **non-modally.**

  Exception for positioning axes: If traverse instruction `POSA` is programmed, the handwheel override can also act modally because this positioning axis does not affect the block transition.

- When the handwheel override is activated with `FDA` or `FD`, a **target position** must be programmed in the NC block for the positioning axis or for a path axis. When the programmed target position is reached, the handwheel override becomes inactive again.
- It is not possible to program `FDA` and `FD` or `FA` and `F` in the same NC block.
- The positioning axis must not be an indexing axis.

Positioning axis

Syntax for handwheel override: `FDA[AXi] = [feedrate value]`

Example 1:

Activate velocity override

```
N10 POS[U]=10 FDA[U]=100 POSA[V]=20 FDA[V]=150 . . .
```

`POS[U]=10`	Target position of positioning axis U
`FDA[U]=100`	Activate velocity override for positioning axis U; axis velocity of U = 100 mm/min
`POSA[V]=20`	Target position of positioning axis V (modally)
`FDA[V]=150`	Activate velocity override for positioning axis V; axis velocity of V = 150 mm/min

Example 2:

Activate path default and velocity override in the same NC block

```
N20 POS[U]=100 FDA[U]= 0 POS[V]=200 FDA[V]=150 . . .
```

`POS[U]=100`	Target position of positioning axis U
`FDA[U]= 0`	Activate path default for positioning axis U;
`POS [V]=200`	Target position of positioning axis V
`FDA[V]=150`	Activate velocity override for positioning axis V; axis velocity of V = 150 mm/min

Path axis

Syntax for handwheel override: `FD = [feedrate value]`

To program "Handwheel override in automatic mode" for path axes, the following preconditions must have been met:

- Active movement commands from group 1: `G01`, `G02`, `G03`, `CIP`
- Exact stop active (`G60`)
- Linear feedrate in mm/min or inch/min active (`G94`)

These preconditions are checked by the control and an alarm is output if any of them is not met.

Example 3:

Activate velocity override

```
N10 G01 X10 Y100 Z200 FD=1500 . . .
```

`X10 Y100 Z200`	Target position of path axes X, Y and Z
`FD=1500`	Activate velocity override for path axes; path velocity = 1500 mm/min

Concurrent positioning axis

The handwheel override for concurrent positioning axes is activated from the PLC via FC18 by setting the appropriate interface signal
DB31, ... DBX62.1 (handwheel override ON).

If the velocity parameter (F_value) is transferred with a value of 0, the activated handwheel override acts as a path default (i.e., in this case the feedrate is not derived from axial machine data
MD32060 $MA_POS_AX_VELO (initial setting for positioning-axis velocity).

References:

- Function Manual, Extended Functions; Positioning Axes (P2)
- Function Manual, Basic Functions, Basic PLC Program (P3)

4.5.3 Special features of handwheel override in automatic mode

Velocity display

The velocity display for handwheel override shows the following values:

- **Set velocity**

  = programmed velocity

- **Actual velocity**

  = resultant velocity including handwheel override

Effect on transverse axes

If the axis is defined as a transverse axis and `DIAMON` is active, the handwheel pulses are interpreted and traversed as diameter values while handwheel override is active.

Dry-run feedrate

With active dry run
DB21, ... DBX0.6 (activate dry-run feedrate) = 1,
the dry-run feedrate is always effective
SD42100 $SC_DRY_RUN_FEED.

In this way, the axis is traversed to the programmed target position at dry-run feedrate without any influence from the handwheel despite the active handwheel override with path default (`FDA[AXi] = 0`), i.e., the path default is ineffective.

DRF active

When "Handwheel override in automatic mode" is activated, it is important to check whether the "DRF" function is active (DB21, ... DBX0.3 = 1).

If this were the case, the handwheel pulses would also cause a DRF offset of the axis. The user must, therefore, first deactivate DRF.

Feedrate override

The feedrate override does not affect the velocity of the movements produced by the handwheel (exception: 0%). It only affects the programmed feedrate.

With path default and fast handwheel movements, the axis may not be able to follow the handwheel rotation synchronously (especially in the case of a large handwheel-pulse weighting), causing the axis to overtravel.

4.6 Contour handwheel/path input using handwheel (option)

Function

When the function is activated, the feedrate of path and synchronized axes can be controlled via a handwheel in AUTOMATIC and MDI modes.

Availability

For the SINUMERIK 840D sl and SINUMERIK 828D systems, the "contour handwheel" function is available as an option that is under license.

Function response

MD11346 $MN_HANDWH_TRUE_DISTANCE

The following response can be set for the contour handwheel using the machine data above (handwheel path or velocity definition):

- **Path definition**

  Limiting the velocity to the maximum permissible value causes the axes to overtravel. The path defined by the handwheel is traversed and **no pulses are lost**.

- **Velocity specification**

  The handwheel only defines the traverse velocity. As soon as the handwheel stops, the axes stop too. Motion is braked immediately if no pulses are supplied from the handwheel in one IPO cycle. thus **preventing overtravel by the axes**. The handwheel pulses do not supply a path default.

Feedrate

The feedrate in mm/min is **dependent** on:

- The number of pulses supplied by the selected handwheel within one period
- The handwheel's pulse weighting set via machine data
  MD11322 $MN_CONTOURHANDWH_IMP_PER_LATCH
  (contour-handwheel pulses per detent position)
- The activated increment (INC1, 10, 100, etc.)
- The path weighting of an increment
  MD31090 $MA_JOG_INCR_WEIGHT
  of the first available geometry axis

The feedrate is **not dependent** on:

- The programmed feedrate mode (mm/min, mm/rev.)
- The programmed feedrate (resultant velocity can be higher)
- The rapid-traverse rate for G0 blocks
- The override (position 0% is effective, i.e., zero speed)

Traversing direction

The traversing direction depends on the direction of rotation:

- **Clockwise**

  → Results in travel in the programmed direction

  If the block-change criterion (IPO end) is reached, the program advances to the next block (response identical to G60).

- **Counterclockwise**

  → Results in travel in the programmed direction

  Here, the axes can only traverse to the appropriate block start. Pulses are not collected if the handwheel continues to rotate.

Activation of the function

The function can be activated via interface signals or via the NC program:

- Activation via interface signal

  The function is activated/deactivated via interface signal
  DB21, ... DBX30.0, 30.1, 30.2 (activate handwheel x as a contour handwheel)

- Activation via the NC program

  The contour handwheel can be activated in the NC program non-modally using FD=0, that is, velocity F from the block before the contour handwheel applies in the following block **without** the need for additional programming.

Note

If no feedrate was programmed in the previous blocks, a corresponding alarm is output.

FD and F cannot appear in the same NC block (triggers an alarm).

Contour-handwheel simulation

When the contour handwheel is activated, it can also be simulated.

After activation via interface signal
DB21, ... DBX30.3 (contour-handwheel simulation),
the feedrate is no longer defined by the contour handwheel; the programmed feedrate is used instead.

The direction is also defined via an interface signal:
DB21, ... DBX30.4 (contour-handwheel-simulation negative direction)

When the simulation is deselected or the direction is changed, the current movement is decelerated using a braking ramp.

Note

The override is effective as for NC-program execution.

Secondary conditions

- **Preconditions**

  Fixed feedrate, dry-run feedrate, thread cutting, or tapping must not be selected.

- **Limit values**

  The acceleration and velocity of the axes are limited to the values defined in the machine data.

- **Interruption of traversing movement**

  On NC Stop, the function remains selected but the handwheel pulses are not summated and are ineffective.

  Precondition: MD32084 $MA_HANDWH_CHAN_STOP_COND bit 2 = 1

  DRF

  A selected DRF function also has a path-override action.

- **Channel-specific deletion distance-to-go**

  This causes the movement triggered by the contour handwheel to be aborted; the axes are decelerated and the program is restarted with the next NC block. The contour handwheel then becomes effective again.

4.7 DRF offset

Function

The "DRF offset" function (differential resolver function) can be used to set an additive incremental work offset in respect of geometry and auxiliary axes in the basic coordinate system in AUTOMATIC mode via an electronic handwheel.

The handwheel assignment, i.e., the assignment of the handwheel from which the increments for the DRF offset are to be derived, to the geometry or auxiliary axes that are to be moved by this, must be performed via the appropriate machine axes. The appropriate machine axes are those machine axes to which the geometry or auxiliary axis is mapped.

The DRF offset is not displayed in the axis actual-value display.

References:
/BAD/ HMI Advanced User Manual
/BEM/ HMI Embedded User Manual

Applications

The DRF offset can be used, for example, in the following application cases:

- Offsetting tool wear within an NC block

  Where NC blocks have very long processing times, it becomes necessary to offset tool wear manually within the NC block (e.g., large surface-milling machines).
- Highly precise offset during grinding
- Simple temperature compensation

⚠ CAUTION
The work offset introduced via the DRF offset is always effective in all modes and after a RESET. It can, however, be suppressed non-modally in the part program.

Velocity reduction

The velocity generated by the handwheel for DRF can be reduced as compared to the JOG velocity via axial machine data
MD32090 $MA_HANDWH_VELO_OVERLAY_FACTOR
(ratio of JOG velocity to handwheel velocity).

DRF active

DRF must be active to allow the DRF offset to be modified by means of traversal with the handwheel. The following preconditions must be fulfilled:

- AUTOMATIC mode
- DB21, ... DBX0.3 (activate DRF) = 1

The DRF offset can be activated/deactivated for specific channels using the "program control" function on the HMI user interface.

The HMI software then sets the interface signal:
DB21, ... DBX24.3 (DRF selected) =1

The PLC program (basic PLC program or user program) transfers this interface signal to interface signal
DB21, ... DBX0.3 (activate DRF) once the corresponding logic operation has been performed.

Control of DRF offset

The DRF offset can be modified, deleted or read:

User:	• Traversing with the handwheel
Part program:	• Reading via axis-specific system variable $AC_DRF[<axis>] • Deleting via parts-program command (DRFOF) for all axes in a channel • Non-modal suppression via parts-program command (SUPA) **References:** /PG/ Programming Manual Fundamentals
PLC user program:	• Reading the DRF offset (axis-specific) **References:** /FB1/ Function Manual, Basic Functions; Basic PLC Program (P3)
HMI user interface:	• Display of the DRF offset (axis-specific)

Note

If DRF offset is deleted, the axis is not traversed!

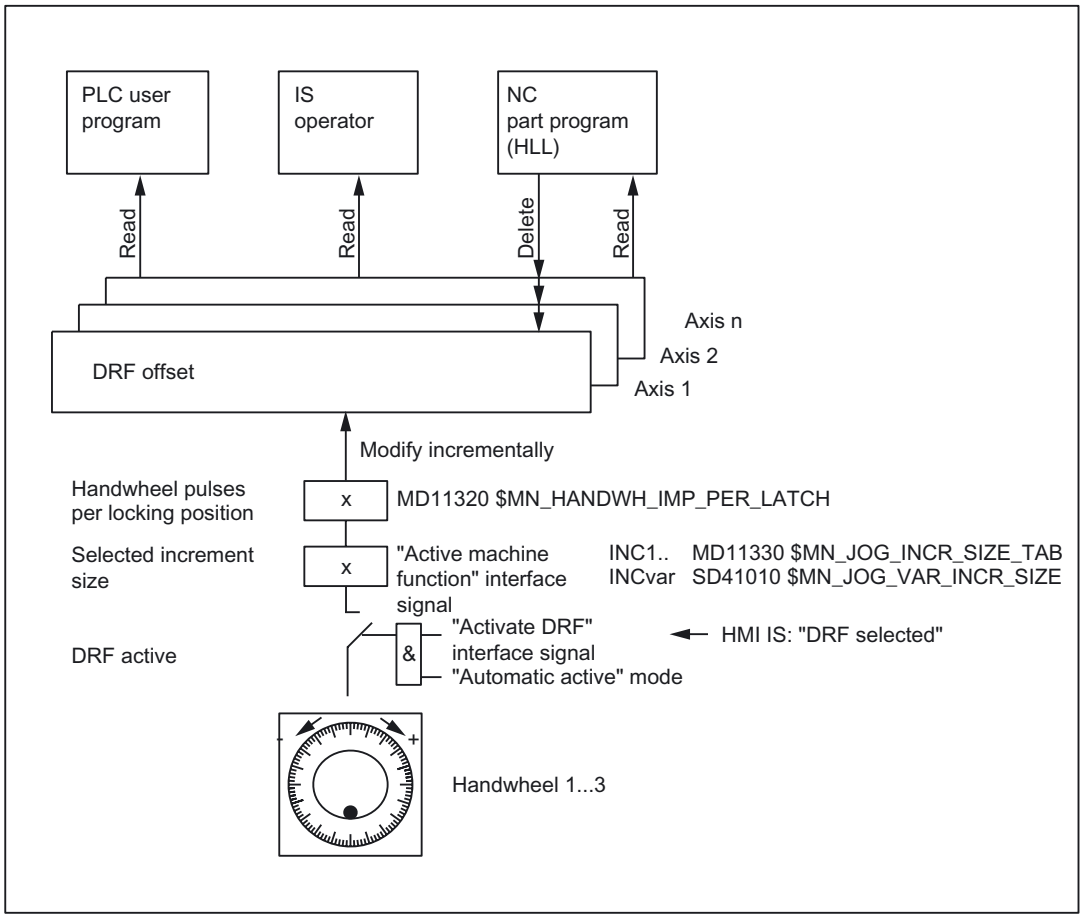


Figure 4-6 Control of DRF offset

Display

The axis actual-position display (ACTUAL POSITION) does not change while an axis is being traversed with the handwheel via DRF. The current axis DRF offset can be displayed in the DRF window.

Reference point approach

In phase 1 of the machine-axis reference point approach, the DRF offset for the corresponding geometry or auxiliary axis is deleted.

During the machine-axis reference point approach, a DRF offset for the corresponding geometry or auxiliary axis cannot be performed simultaneously.

Reset response

`PowerOn-Reset`: The DRF offset is deleted.

4.8 Start-up: Handwheels

4.8.1 General information

In order to operate handwheels of a SINUMERIK control system, they have to be parameterized via NCK machine data.

If the handwheels are not connected to the control directly, other measures must be taken, e.g. in case of connection through PROFIBUS-MCP or handwheel module, the insertion and configuration of the module with SIMATIC STEP 7, HW Config.

Note

Currently only 6 handwheels can be parameterized in a SINUMERIK control system.

Connection options

SINUMERIK 840D sl

For SINUMERIK 840D sl, handwheels can be connected via the following components:

- PROFIBUS Module
- Ethernet Module

Note

Several handwheels, which are connected via different components, can be connected to a SINUMERIK 840D slcontrol system simultaneously.

SINUMERIK 828D

For SINUMERIK 828D, handwheels are directly connected to terminal X143 of the PPU.

Note

A maximum of 2 handwheels can be connected to SINUMERIK 828D control system.

4.8.2 Connection via PPU - only 828D

Parameterization

Handwheels directly connected to terminal X143 of the PPU are parameterized using the following NCK machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[<*Handwheel_No._in_NCK - 1*>] = 2

  When directly connected to the PPU, a 2 must always be entered as hardware segment.
- MD11351 $MN_HANDWHEEL_MODULE[< *Handwheel_No._in_NCK - 1* >] = 1

  When directly connected to the PPU, a 1 must always be entered as hardware module.
- MD11352 $MN_HANDWHEEL_INPUT[< *Handwheel_No_in_NCK - 1* >] = <*Handwheel connection* >

  Handwheel connection used: 1 or 2

Note

A maximum of 2 electronic handwheels can be connected to terminal X143 of the PPU.

Example

Parameterizing 2 handwheels that are directly connected to the PPU via terminal X143.

Machine data	Value	Significance
		1st handwheel:
MD11350 $MN_HANDWHEEL_SEGMENT[0]	2	Always 2 when connected via PPU
MD11351 $MN_HANDWHEEL_MODULE[0]	1	Always 1 when connected via PPU
MD11352 $MN_HANDWHEEL_INPUT[0]	1	1st handwheel connected to PPU
		2nd handwheel:
MD11350 $MN_HANDWHEEL_SEGMENT[1]	2	Always 2 when connected via PPU
MD11351 $MN_HANDWHEEL_MODULE[1]	1	Always 1 when connected via PPU
MD11352 $MN_HANDWHEEL_INPUT[1]	2	2nd handwheel connected to PPU

4.8.3 Connected via PROFIBUS - only 840D sl

Parameter setting

Parameterization of handwheels connected via PROFIBUSmodules (e.g. machine control table "MCP 483") is done with the following NCK machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[<*Handwheel_No._in_NCK - 1*>] = 5

  When connected via PROFIBUSmodule, the hardware segment has always to be entered as 5 (PROFIBUS).

- MD11351 $MN_HANDWHEEL_MODULE[<*Handwheel_No._in_NCK - 1*>] = <*Index + 1*>

  The reference to the MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS[<*Index*>]

  has to be entered, which contains the logical base address of the handwheel.

- MD11352 $MN_HANDWHEEL_INPUT[<*Handwheel_No._in_NCK - 1*>] = <*Number_in_handwheel_slot*>

  A handwheel slot can contain several handwheels. The number of the handwheel within the handwheel slot has to entered: 1, 2, ...

- MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS[<*Index*>] = <*logical base address*>

  The logical base address of the handwheel slot, specified in SIMATIC STEP 7, HW config, has to be entered.

Handwheel slot

The PROFIBUSmodule must be configured besides the parameterization of handwheels in the NCK machine data in STEP 7.Among others the logical address of the handwheel slot is specified.

The handwheel slot is situated at the following slot of the PROFIBUSmodule:

PROFIBUS module	Slot
Machine control panel MCP 438	2
Machine control panel MCP 310	2
Handwheel connection module	1

Example

Parameterization of 5 handwheels, connected via 4 machine control tables "MCP 483". Two handwheels can be connected to a machine control table "MCP 483".

Handwheel number in NCK	Machine data set (Index)	Connection
1	0	1st MCP, 1st handwheel in handwheel slot
2	1	1st MCP, 2nd handwheel in handwheel slot
3	2	2nd MCP, 1st handwheel in handwheel slot
5	4	3rd MCP, 1st handwheel in handwheel slot
6	5	4th MCP, 2nd handwheel in handwheel slot

The fourth handwheel in NCK is not used (gap in machine data).

Note

Machine data gaps are allowed when parameterizing handwheels in NCK machine data.

Machine control tables have been configured in SIMATIC STEP 7, HW Config as follows:

	Slot	DP ID	Order No. / Description ...	I address	O address
1st MCP	1	55	Standard+Handwheel	0 ... 7	0 ... 7
	2	*2AE*	*→ standard+handwheel*	*288 ... 291*	
	3	*1*	*→ standard+handwheel*		
2nd MCP	1	55	Standard+Handwheel	8 ... 15	8 ... 15
	2	*2AE*	*→ standard+handwheel*	*304 ... 307*	
	3	*1*	*→ standard+handwheel*		
3rd MCP	1	55	Standard+Handwheel	16 ... 23	16 ... 23
	2	*2AE*	*→ standard+handwheel*	*320 ... 323*	
	3	*1*	*→ standard+handwheel*		
4th MCP	1	55	Standard+Handwheel	24 ... 29	24 ... 29
	2	*2AE*	*→ standard+handwheel*	*330 ... 333*	
	3	*1*	*→ standard+handwheel*		

Parameterizing in the NCK machine data:

Machine data	Value	Description
		1st handwheel in NCK
MD11350 $MN_HANDWHEEL_SEGMENT[0]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[0]	1	Reference to logical base address of the handwheel slot of the 1st MCP
MD11352 $MN_HANDWHEEL_INPUT[0]	1	1st handwheel in handwheel slot
		2nd handwheel in NCK
MD11350 $MN_HANDWHEEL_SEGMENT[1]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[1]	1	Reference to logical base address of the handwheel slot of the 1st MCP
MD11352 $MN_HANDWHEEL_INPUT[1]	2	2nd handwheel in handwheel slot
		3rd handwheel in NCK
MD11350 $MN_HANDWHEEL_SEGMENT[2]	5	Hardware segment: PROFIBUS

Machine data	Value	Description
MD11351 $MN_HANDWHEEL_MODULE[2]	2	Reference to logical base address of the handwheel slot of the 2nd MCP
MD11352 $MN_HANDWHEEL_INPUT[2]	1	1st handwheel in handwheel slot
		4th handwheel in NCK
MD11350 $MN_HANDWHEEL_SEGMENT[3]	0	No handwheel parameterized
MD11351 $MN_HANDWHEEL_MODULE[3]	0	No handwheel parameterized
MD11352 $MN_HANDWHEEL_INPUT[3]	0	No handwheel parameterized
		5th handwheel in NCK
MD11350 $MN_HANDWHEEL_SEGMENT[4]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[4]	6	Reference to logical base address of the handwheel slot of the 3rd MCP
MD11352 $MN_HANDWHEEL_INPUT[4]	1	1st handwheel in handwheel slot
		6th handwheel in NCK
MD11350 $MN_HANDWHEEL_SEGMENT[5]	5	Hardware segment: PROFIBUS
MD11351 $MN_HANDWHEEL_MODULE[5]	5	Reference to logical base address of the handwheel slot of the 4th MCP
MD11352 $MN_HANDWHEEL_INPUT[5]	2	2nd handwheel in handwheel slot

Logical base addresses:

Machine data	Value	Description
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [0]	288	Logical base address handwheel slot 1st MCP
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [1]	304	Logical base address handwheel slot 2nd MCP
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [4]	330	Logical base address handwheel slot 4th MCP
MD11353 $MN_HANDWHEEL_LOGIC_ADDRESS [5]	320	Logical base address handwheel slot 3rd MCP

4.8.4 Connected via Ethernet - only 840D sl

Parameter setting

The parameters for handwheels connected via Ethernet modules, e.g. machine control panel "MCP 483C IE", "HT 8", or "HT 2", are assigned in the following NC machine data:

- MD11350 $MN_HANDWHEEL_SEGMENT[< x - 1 >] = 7

  When connected via Ethernet modules, the segment always has to be entered as 7 (Ethernet).

- MD11351 $MN_HANDWHEEL_MODULE[< x - 1 >] = 1

  When connected via Ethernet modules, the module always has to be entered as 1.

- MD11352 $MN_HANDWHEEL_INPUT[< x - 1 >] = y

  where y = 1, 2, 3, etc. (handwheel interface at the Ethernet bus)

where x = 1, 2, 3, etc. (handwheel number in the NC)

Handwheel interfaces at the Ethernet Bus

The handwheel interfaces at the Ethernet bus are numbered on the basis of the following considerations:

- The sequence of the operator component interfaces is: MCP1, MCP2, BHG
- Each operator component interface has two handwheel interfaces
- Operator components: MCP 483C IE

  A maximum of two handwheels can be connected to an MCP 483C IE via connections X60 and X61 . The assignment of the connections in the operator component interface is:

  – Connection X60: 1st handwheel in operator component interface MCP1 /MCP2

  – Connection X61: 2nd handwheel in operator component interface MCP1 /MCP2
- Operator components: HT 8

  The handwheel of the HT 8 is always assigned to the 1st handwheel of operator component interface MCP1 /MCP2.
- Operator components: HT 2

  The handwheel of the HT 2 is always assigned to the 1st handwheel of operator component interface BHG.

Operator component interface ->	**MCP1**		**MCP2**		**BHG**	
Handwheel interface [1]	1	2	1	2	1	2
FB1 parameters [2]	MCP1BusAdr		MCP2BusAdr		BHGRecGDNo	
Assignment of the handwheels [3]						
MCP 483C IE	X60	X61	X60	X61	-	-
HT 8	x	-	x	-	-	-
HT 2	-	-	-	-	x	-
Handwheel interface at the Ethernet bus (y) [4] ->	1	2	3	4	5	6

1) Numbering of the handwheel interfaces within an operator component interface
2) Assignment of the operator component to the interface via the corresponding FB1 parameter
3) Assignment of the handwheels of the respective operator components to the handwheel interfaces
4) Numbering of the handwheel interfaces at the Ethernet bus -> MD11352 $MN_HANDWHEEL_INPUT[< x - 1 >] = **y**

Example

Parameterization of 3 handwheels, connected via the following operator components:

Operator component interface ->	**MCP1**		**MCP2**		**BHG**	
Operator component	HT 8		MCP 483C		HT 2	
FB1 parameters	MCP1BusAdr := 39		MCP2BusAdr := 192		BHGRecGDNo := 40	
Handwheel interface	**x**	-	-	**X61**	**x**	-
Handwheel interface at Ethernet Bus ->	**1**	2	3	**4**	**5**	6

Table 4- 1 NCK machine data for the handwheel assignment

Machine data	Value	Description
		HT 8: Handwheel number in the NC = 1
MD11350 $MN_HANDWHEEL_SEGMENT[0]	7	Segment: Ethernet
MD11350 $MN_HANDWHEEL_MODULE[0]	1	Module: Ethernet
MD11350 $MN_HANDWHEEL_INPUT[0]	1	Handwheel interface at Ethernet bus
		MCP 483C IE: Handwheel number in the NC = 2
MD11350 $MN_HANDWHEEL_SEGMENT[1]	7	Segment: Ethernet
MD11350 $MN_HANDWHEEL_ MODULE [1]	1	Module: Ethernet
MD11350 $MN_HANDWHEEL_ INPUT [1]	4	Handwheel interface at Ethernet bus
		HT 2: Handwheel number in the NC = 3
MD11350 $MN_HANDWHEEL_SEGMENT[2]	7	Segment: Ethernet
MD11350 $MN_HANDWHEEL_ MODULE [2]	1	Module: Ethernet
MD11350 $MN_HANDWHEEL_ INPUT [2]	5	Handwheel interface at Ethernet bus

Table 4- 2 FB1 parameters (excerpt)

```
Parameter           Value           Remark
MCPNum              := 2            // Number of connected MCP
                                    // MCP1 = HT 8
MCP1In              ...             // MCP1-Parameter ...
...                 ...
MCP1BusAdr          := 39           // Via switches S1 and S2 on the connecting device
                                    // set "IP address"
                                    // MCP2 = MCP 483C IE
MCP2In              ...             // MCP2-Parameter ...
...                 ...
MCP2BusAdr          := 192          // Via switch S2 on the MCP 483C
                                    // set "IP address"
MCPBusType          := b#16#55      // Bus type: Ethernet
                                    // HHU = HT 2
HHU                 := 5            // Bus type: Ethernet
HHUIn               ...             // HHU Parameter ...
...                 ...
HHURecGDNo          := 40           // Via switches S1 and S2 on the connecting device
                                    // set "IP address"
```

Filter time

Since the handwheel pulses on the Ethernet bus are not transferred deterministically, filtering (smoothing) of the handwheel pulse transfer process may be necessary for highly dynamic drives. The parameter for the filter time is assigned using the following machine data:

- MD11354 $MN_HANDWHEEL_FILTER_TIME[< x - 1 >] = <filter time>

  where x = 1, 2, 3, etc. (handwheel number in the NC) and filter time = 0.0 to 2.0 s

The filter time specifies the time it takes for the handwheel pulses transferred to the control to be sent on to the interpolator for traversing purposes. With a filter time of 0.0 s, the handwheel pulses are sent on to the interpolator within a single interpolation cycle. This can result in the relevant axis being traversed jerkily.

The recommended filter time is 0.2 to 0.5 s.

Stationary state detection

A stationary state is detected by the Ethernet modules to which the handwheel is connected. If a handwheel does not transfer any handwheel pulses for a defined period of time, the module detects this to be a stationary state and transfers it to the NC/PLC interface:

NC/PLC interface signal	Value	Description
DB10, DBX245.0	0	Handwheel 1 is operated
	1	Handwheel 1 is stationary
DB10, DBX245.1	0	Handwheel 2 is operated
	1	Handwheel 2 is stationary
DB10, DBX245.2	0	Handwheel 3 is operated
	1	Handwheel 3 is stationary

By evaluating the signal, it is possible to reduce the overtravel of an axis traversed via the handwheel, due to the handwheel pulses that have been collected in the control but not yet transferred to the interpolator for traversing purposes. To do this, deletion of the distance-to-go must be triggered for the relevant axis or in the channel when a stationary state is detected:

- DB31,... DBX2.2 = 1 (axial deletion of distance-to-go)
- DB21,... DBX6.2 = 1 (channel-spec. deletion of distance-to-go)

4.9 Special features of manual travel

4.9.1 Geometry-axis manual travel

Coordinate systems in JOG

In JOG mode, the user can also traverse the axes declared as geometry axes in the workpiece coordinate system manually. Any coordinate offsets or rotations that have been selected remain active.

Note

With SINUMERIK 840D, the "handling transformation package" can be used in JOG mode to set the translation of geometry axes separately in several valid reference systems.

References:
/FB3/ Function Manual Special Functions; 3- to 5-Axis Transformation (F2), Chapter: "Cartesian manual travel"

Application

Manual movements for which transformations and frames have to be active. The geometry axes are traversed in the most recently valid coordinate system. The special features of geometry-axis manual travel are described below.

Simultaneous travel

Only one geometry axis can be traversed continuously or incrementally at one time using the traversing keys. Where an attempt is made to traverse more than one geometry axis, alarm 20062 "Axis already active" is output. However, three geometry axes can be traversed simultaneously using handwheels 1, 2 and 3. Alarm 20060 is output if only one axis is not defined as a geometry axis.

PLC interface

A separate PLC interface
(DB21, ... DBB12 - 23 and DBB40 - 56)
exists for geometry axes, which contains the same signals as the axis-specific PLC interface.

Feedrate/rapid traverse override

The channel-specific feedrate-override switch and rapid-traverse-override switch are active for geometry-axis manual travel in rapid traverse override.

Alarms

Alarm 20062, "Axis already active", is triggered in the case of geometry-axis manual travel under the following conditions:

- The axis is already being traversed in JOG mode via the axial PLC interface.
- A frame for a rotated coordinate system is already active and another geometry axis in this coordinate system is traversed in JOG mode with the traversing keys.

If the axis is not defined as a geometry axis, alarm 20060, "Axis cannot be traversed as a geometry axis", is output if you attempt to traverse it as a geometry axis in JOG mode.

4.9.2 Special features of spindle manual travel

Spindle manual travel

Spindles can also be traversed manually in JOG mode. Essentially, the same conditions apply as for manual travel of axes. Spindles can be traversed in JOG mode using the traversing keys continuously or incrementally, in jog or continuous mode, or using the handwheel. The function is selected and activated via the axis-/spindle-specific PLC interface in the same way as for the machine axes. The axis-specific machine data also apply to the spindles.

Spindle mode

Spindle manual travel is possible in positioning mode (spindle is in position control) or in open-loop control mode.

JOG velocity

The velocity used for spindle manual travel can be defined as follows:

- Using general setting data
  SD41200 $SN_JOG_SPIND_SET_VELO (speed of spindle in JOG mode), which is valid for all spindles

or

- Using machine data
  MD32020 $_MA_JOG_VELO (JOG axis velocity)

  However, the machine data is only effective if
  SD41110 $SN_JOG_SET_VELO (axis velocity in JOG) = 0.

The maximum speeds for the active gear stage also apply when spindles are traversed in JOG mode.

References:
/FB1/ Function Manual Basic Machine; Spindles (S1)

Velocity override

The spindle-override-switch JOG velocity is active for spindles.

JOG acceleration

As a spindle often uses many gear stages in speed-control and position-control modes, the acceleration associated with the current gear stage is always applied to the spindle in JOG mode.

References:
/FB1/ Function Manual Basic Machine; Spindles (S1)

PLC interface signals

In the case of spindle manual travel, the PLC interface signals between the NCK and PLC have the same effect as for machine axes.

Interface signals
DB31, ... DBX60.7 or DBX60.6 (position reached with fine or coarse exact stop)
are only set if the spindle is in position control.

In the case of interface signals that are only spindle-specific, while the spindles are traversing in JOG the following should be noted:

- The following PLC interface signals to the spindle have no effect:
  - DB31, ... DBX17.6 (invert M3/M4)
  - DB31, ... DBX18.7 or DBX18.6 (counterclockwise setpoint direction or clockwise setpoint direction)
  - DB31, ... DBX18.5 (oscillation speed)
  - DB31, ... DBX16.7 (spindle RESET)
- The following PLC interface signals from the spindle are not set:
  - DB31, ... DBX83.7 (clockwise actual direction)
  - DB31, ... DBX83.5 (spindle in set range)

4.9.3 Monitoring functions

Limitations

The following limitations are active for manual travel:

- Working-area limitation (axis must be referenced)
- Software limit switches 1 and 2 (axis must be referenced)
- Hardware limit switches

The control ensures that the traversing movement is aborted as soon as the first valid limitation has been reached. Velocity control ensures that deceleration is initiated early enough for the axis to stop exactly at the limit position (e.g., software limit switch). Only when the hardware limit switch is triggered does the axis stop abruptly with "rapid stop".

Alarms are triggered when the various limitations are reached (alarms 16016, 16017, 16020, 16021). The control automatically prevents further movement in this direction. The traversing keys and the handwheel have no effect in this direction.

Note

The software limit switches and working-area limitations are only active if the axis has first been referenced.

If a work offset (DRF offset) via handwheel is active for axes, the software limit switches for these axes are monitored during the main run in JOG mode. This means that the jerk limitation has no effect when the software limit switches are approached. The velocity is reduced at the software limit switch after the acceleration
(MD32300 $MA_MAX_AX_ACCEL (axis acceleration))
has been performed.

For further information on working-area limitations and hardware and software limit switches, see:
References:
/FB1/ Function Manual, Axis Monitoring, Protection Zones (A3)

Retract axis

The axis can be retracted from a limit position by moving it in the opposite direction.

Note

Machine manufacturer

The function for retracting an axis that has approached the limit position depends on the machine manufacturer. Please refer to the machine manufacturer's documentation!

Maximum velocity and acceleration

The velocity and acceleration used during manual travel are defined by the startup engineer for specific axes using machine data. The control limits the values acting on the axes to the maximum velocity and acceleration specifications.

References:
/FB1/ Function Manual Basic Machine; Velocities, Setpoint/Actual-Value Systems, Closed-Loop Controls (G2)
/FB1/ Function Manual Basic Machine; Acceleration (B2)

4.9.4 Other

Mode change from JOG to AUT or from JOG to MDI

It is possible to switch operating modes from JOG to AUT or MDI only if all axes in the channel have reached "coarse exact stop".

References:
/FB1/ Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1)

Rotational feedrate active in JOG

In JOG mode, it is also possible to traverse an axis manually at a revolutional feedrate (as for `G95`) specific to the current speed of the master spindle.

It is activated via setting data
SD41100 **$SN_**JOG_REV_IS_ACTIVE (JOG: revolutional/linear feedrate).

The feedrate value (in mm/rev) used can be defined as follows:

- Using general setting data
  SD41120 $SN_JOG_REV_SET_VELO (JOG velocity for G95)
- Via axial machine data
  MD32050 $MA_JOG_REV_VELO (revolutional feedrate in JOG)
  or for rapid traverse override:
  MD32040 $MA_JOG_REV_VELO_RAPID
  (revolutional feedrate in JOG with rapid traverse override),
  if SD41120 = 0

If a master spindle has not been defined and the axis is to be traversed in JOG at a revolutional feedrate, alarm 20055 is output (alarm 20065 for geometry axes).

Transverse axes

If a geometry axis is defined as a transverse axis and radius programming is selected (MD20100 $MC_DIAMETER_AX_DEF (geometry axes with transverse-axis functions)), the following features should be observed when traversing in JOG:

- Continuous travel:

  There are no differences when a transverse axis is traversed continuously.

- Incremental travel:

  Only **half the distance** of the selected increment size is traversed. For example, with INC10 the axis only traverses 5 increments when the traversing key is pressed.

- Traversing with the handwheel:

  As for incremental travel, with the handwheel only half the distance is traversed per handwheel pulse.

References:
/FB1/ Function Manual, Basic Functions; Transverse Axes (P1)

4.10 Approaching a fixed point in JOG

4.10.1 Introduction

Function

The machine user can use the "Approaching fixed point in JOG" function to approach axis positions defined through machine data by actuating the traverse keys of the machine control table. The traveling axis comes to a standstill automatically on reaching the defined fixed point.

Applications

Typical applications are, for example:

- Approaching a basic position before starting an NC program.
- Travel towards tool change points, loading points and pallet change points.

Requirements

- The "Approaching fixed point in JOG" can be activated only in the JOG mode.

  The function cannot be enabled in the JOG-REPOS and JOG-REF sub-modes and in JOG in the AUTOMATIC mode.
- The axis to be traversed must be referenced.
- A kinematic transformation may not be active.
- The axis to be traversed may not be a following axis of an active coupling.

Approaching fixed point with G75/G751

Approaching of the defined fixed points can be activated from the part program too using the `G75`/`G751` command.

For more information on approaching fixed points with `G75`/`G751`, please refer to:
References:
Programming Manual Fundamentals; Section "Additional commands" > "Approaching fixed point (G75, G751)".

4.10.2 Functionality

Procedure

Procedure in "Approaching fixed point in JOG"

- Selection of JOG mode
- Enabling the "Approach fixed point in JOG" function
- Traversing of the machine axis with traverse keys or handwheel

Activation

The PLC sets the interface signal after the "Approach fixed point in JOG" function has been selected:

DB31, ... DBX13.0-2 (JOG - Approach fixed point)

The number of the approaching fixed point is to output through Bit 0 - 2 binary code.

The NC confirms the activation with the interface signal as soon as the function is effective:

DB31, ... DBX75.0-2 (JOG - Approach fixed point)

Note

Activation is not possible:

- during an NCK reset
- In case of impending emergency stop
- During processing of an ASUP

No alarm message occurs. Delayed activation takes place after closure or after acknowledgement of the active function.

Sequence

The actual traversing is started with the traverse keys or the handwheel in the direction of the approaching fixed point.

The selected machine axis traverses till it comes to an automatic standstill at the fixed point.

The corresponding NC/PLC interface signal is reported on reaching the fixed point with "Exact stop fine":

DB31, ... DBX75.3-5 (JOG - Approach fixed point)

This display signal is also signaled if the axis reaches the fixed point position in the machine coordinates system via other methods e.g. NC program, FC18 (for 840D sl) or synchronized action on the setpoint side and comes to a standstill on the actual value side within the "Exact stop fine" tolerance window (MD36010 $MA_STOP_LIMIT_FINE).

Movement in the opposite direction

The response while traversing in the opposite direction, i.e.,against the direction of the approaching fixed point depends on the setting of Bit 2 in the machine data:

MD10735 $MN_JOG_MODE_MASK (settings for the JOG mode)

Traverse in the opposite direction is possible only if the bit is set.

Traverse in the opposite direction is blocked if the bit is not set and the following channel status message is output if an attempt is made with the traverse keys or with the handwheel to traverse in the direction opposite the approaching fixed point:

"JOG: <Axis> direction blocked"

Approaching other fixed point

The axis motion is cancelled and the following alarm is output if a different fixed point is selected while traversing to the fixed point:

Alarm 17812 "Channel %1 Axis %2 fixed point approach in JOG: Fixed point changed"

The message signal DB31, ... DBX75.0-2 (JOG - Approaching fixed point active) displays the number of the newly selected fixed point. The JOG traverse must be triggered again to continue traversing.

Note

To avoid the alarm message, the machine user should proceed as follows:

1. Cancel the current traverse movement with residual distance deletion.
2. Activate fixed point approach for another fixed point and start the operation after the axis comes to a standstill.

Withdrawal from fixed point/deactivation

To withdraw from a fixed position, you must deactivate the "Approaching fixed point in JOG" function. This is done by resetting the activation signal to "0".

DB31, ... DBX13.0-2 = 0

The message signals DB31, ... DBX75.0-2 (JOG - Approaching fixed point active) and DB31, ... DBX75.3-5 (JOG - Approaching fixed point reached) are deleted on leaving the fixed point position.

Special case: Axis is already on fixed point

The axis cannot be moved if, while starting the fixed point traverse, the axis is already at the position of the fixed point to be approached. This is displayed through the following channel status message:

"JOG: <Axis> position reached"

To withdraw from the fixed position, you must deactivate the "Approaching fixed point in JOG" function.

Special features of incremental travel

If, during incremental travel, the fixed point is reached before the increment is completed, then the increment is considered to have been completed fully. This is the case even when only whole increments are traveled.

MD11346 $MN_HANDWH_TRUE_DISTANCE = 2 or 3

Features of modulo rotary axes

Modulo rotary axes can approach the fixed point in both directions. The shortest path (DC) is not observed during the travel.

Features of spindles

A spindle changes to the positioning mode on actuating the "Approaching fixed point in JOG" function. The closed loop position control is active and the axis can traverse to the fixed point.

If no zero mark is detected the alarm message in the axis operation is output:

Alarm 17810 "Channel %1 Axis %2 not referenced"

As a spindle must also be a modulo rotary axis at all times, the same conditions apply for direction observation as for modulo rotary axes (refer to the paragraph "Features of modulo rotary axes")

4.10.3 Parameter setting

Movement in the opposite direction

The response while traversing in the opposite direction, i.e.,against the direction of the approaching fixed point depends on the setting of Bit 2 in the machine data:

MD10735 $MN_JOG_MODE_MASK (settings for the JOG mode)

Bit	Value	Description
2	0	Travel in the opposite direction is **not** possible (default setting).
	1	Movement in the opposite direction is possible.

Fixed point positions

A maximum of 4 fixed point positions can be defined for each axis via the following machine data:

MD30600 $MA_FIX_POINT_POS[n]

Number of valid fixed point positions

The number of valid fixed point positions of an axis is defined via the machine data:

MD30610 $MA_NUM_FIX_POINT_POS

Note

"Approaching fixed point with `G75`/`G751`" constitutes an exception in this case. Two fixed point positions can also be approached here in the setting MD30610 = 0.

4.10.4 Programming

System variables

The following system variables that can be read in the part program and in the synchronous actions for the "Approach fixed point" function.

System variable	Description
$AA_FIX_POINT_SELECTED [<Axis>]	Number of fixed point to be approached
$AA_FIX_POINT_ACT [<Axis>]	Number of the fixed point on which the axis is currently located

4.10.5 Supplementary conditions

Axis is indexing axis

The axis is not traversed and an alarm is output if the axis to be traversed is an indexing axis and the fixed point position to be approached does not match an indexing position.

Frames active

All active frames are ignored. Traversing is performed in the machine coordinate system.

Offset values active

Active offset values (DRF, external zero offset, synchronized action offset $AA_OFF, online tool offset) are also traversed. The fixed point is a position in the machine coordinates system.

An alarm is issued if an offset movement (DRF, external zero offset, synchronized action offset $AA_OFF, online tool offset) is made during a fixed point approach in JOG. The position of the fixed point to be approached in the machine coordinates system is not reached; instead a position that would have been reached without active offset movement is reached. The NC/PLC interface signal DB31, ... DBX75.3-5 corresponding to the fixed point is not output.

Working-area limitations

Working-area limitations (in BCS and WCS) are considered and the axis motion is stopped on reaching the limits.

4.10.6 Application example

Target

A rotary axis (machine axis 4 [AX4]) is to be moved to Fixed Point 2 (90 degrees) with the "Approaching fixed point in JOG" function.

Parameter setting

The machine data for the "Approaching fixed point" function of machine axis 4 are parameterized as follows:

MD30610 $MA_NUM_FIX_POINT_POS[AX4] = 4	4 fixed points are defined for machine axis 4.
MD30600 $MA_FIX_POINT_POS[0,AX4] = 0	1st fixed point of AX4 = 0 degree
MD30600 $MA_FIX_POINT_POS[1,AX4] = 90	2nd fixed point of AX4 = 90 degree
MD30600 $MA_FIX_POINT_POS[2,AX4] = 180	3rd fixed point of AX4 = 180 degree
MD30600 $MA_FIX_POINT_POS[3,AX4] = 270	4th fixed point of AX4 = 270 degree

Initial situation

Machine axis 4 is referred and is in Position 0 degree. This corresponds to the 1st fixed position and is output through the NC/PLC interface signal:

DB31 DBX75.0 = 1 (Bit 0-2 = 1)

Approaching fixed point 2

The control is switched in the JOG mode.

The "Approaching fixed point" function is activated on Fixed Point 2 via the NC/PLC interface signal:

DB31 DBX13.1 = 1 (Bit 0-2 = 2)

The actuation is confirmed via the NC/PLC interface signal:

DB31 DB75.1 = 1 (Bit 0-2 = 2)

The Plus traverse key in the machine control table is used to traverse continuously to approach Fixed Point 2.

The machine axis 4 stops at the 90 degree position. This is reported via the NC/PLC interface signal:

DB31 DBX75.4 = 1 (Bit 3-5 = 2)

4.11 Data lists

4.11.1 Machine data

4.11.1.1 General machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB[n]	Machine axis name
10735	JOG_MODE_MASK	Settings of the JOG mode
11300	JOG_INC_MODE_LEVELTRIGGRD	INC and REF in inching mode
11310	HANDWH_REVERSE	Defines movement in the opposite direction
11320	HANDWH_IMP_PER_LATCH[n]	Handwheel pulses per locking position
11324	HANDWH_VDI_REPRESENTATION	Coding of handwheel number (NCK/PLC interface)
11330	JOG_INCR_SIZE_TAB[n]	Increment size for INC/handwheel
11340	ENC_HANDWHEEL_SEGMENT_NR	Third handwheel: bus segment
11342	ENC_HANDWHEEL_MODULE_NR	Third handwheel: Logical drive number
11344	ENC_HANDWHEEL_INPUT_NR	Third handwheel: Encoder interface
11346	HANDWH_TRUE_DISTANCE	Handwheel path or velocity specification
11350	HANDWHEEL_SEGMENT[n]	Handwheel segment
11351	HANDWHEEL_MODULE[n]	Handwheel module
11352	HANDWHEEL_INPUT[n]	Handwheel connection
11353	HANDWHEEL_LOGIC_ADDRESS[n]	Logical handwheel slot address (STEP 7)
17900	VDI_FUNCTION_MASK	Function mask for VDI signals

4.11.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20060	AXCONF_GEOAX_NAME_TAB	Geometry axis in channel [n = geometry axis number]
20100	DIAMETER_AX_DEF	Geometry axes with transverse axis functions
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
20620	HANDWH_GEOAX_MAX_INCR_SIZE	Limitation of the geometry axes
20622	HANDWH_GEOAX_MAX_INCR_VSIZE	Path-velocity override
20624	HANDWH_CHAN_STOP_COND	Diverse parameters for handwheel travel
21150	JOG_VELO_RAPID_ORI	Conventional rapid traverse for orientation axes
21165	JOG_VELO_GEO	Conventional speed for geometry axes

4.11.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30450	IS_CONCURRENT_POS_AX	Default setting at reset: neutral axis or channel axis
30600	FIX_POINT_POS[n]	Fixed point positions of the axis
30610	NUM_FIX_POINT_POS	Number of fixed point positions of an axis
31090	JOG_INCR_WEIGHT	Weighting of an increment for INC/handwheel
32000	MAX_AX_VELO	Maximum axis velocity
32010	JOG_VELO_RAPID	Rapid traverse in jog mode
32020	JOG_VELO	JOG axis velocity
32040	JOG_REV_VELO_RAPID	Revolutional feedrate in JOG mode with rapid traverse override
32050	JOG_REV_VELO	Revolutional feedrate for JOG
32060	POS_AX_VELO	Initial setting for positioning-axis velocity
32080	HANDWH_MAX_INCR_SIZE	Limitation of the size of the selected increment
32082	HANDWH_MAX_INCR_VELO_SIZE	Limitation of selected increment for velocity override
32084	HANDWH_STOP_COND	Effect of axis-specific VDI interface signals bits 0...5 on the handwheel
32090	HANDWH_VELO_OVERLAY_FACTOR	Ratio of JOG velocity to handwheel velocity (with DRF)
32300	MAX_AX_ACCEL	Maximum axis acceleration
32430	JOG_AND_POS_MAX_JERK	Max. axial jerk for JOG and POS
35130	GEAR_STEP_MAX_VELO_LIMIT[n]	Maximum velocity for gear stage

4.11.2 Setting data

4.11.2.1 General setting data

Number	Identifier: $SN_	Description
41010	JOG_VAR_INCR_SIZE	Size of variable increment for INC/handwheel
41050	JOG_CONT_MODE_LEVELTRIGGRD	JOG continuous in inching mode
41100	JOG_REV_IS_ACTIVE	Revolutional feedrate in JOG mode active
41110	JOG_SET_VELO	JOG velocity for linear axes (for G94)
41120	JOG_REV_SET_VELO	JOG velocity (for G95)
41130	JOG_ROT_AX_SET_VELO	JOG velocity for rotary axes
41200	JOG_SPIND_SET_VELO	JOG velocity for the spindle

4.11.3 Signals

4.11.3.1 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Handwheel 1 is operated	DB10.DBB68	DB2700.DBB12
Handwheel 2 is operated	DB10.DBB69	DB2700.DBB13
Handwheel 3 is operated	DB10.DBB70	-
Channel number for geometry axis, handwheel 1, 2, 3	DB10.DBB97/98/99	DB1900.DBB1003
Axis number for handwheel 1, 2, 3, handwheel selected and machine axis	DB10.DBB100/101/102	DB1900.DBB1004

4.11.3.2 Signals to mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
JOG mode	DB11.DBX0.2	DB3000.DBX0.2

4.11.3.3 Signals from mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Active JOG mode	DB11.DBX4.2	DB3100.DBX0.2

4.11.3.4 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Activate DRF	DB21, ...DBX0.3	DB3200.DBX0.3
Activate handwheel (3, 2, 1)	DB21, ...DBX12.0-2 DB21, ...DBX16.0-2 DB21, ...DBX20.0-2	DB3200.DBX1000.0-1 DB3200.DBX1004.0-1 DB3200.DBX1008.0-1
Traversing key lock	DB21, ...DBX12.4 DB21, ...DBX16.4 DB21, ...DBX20.4	DB3200.DBX1000.4 DB3200.DBX1004.4 DB3200.DBX1008.4
Rapid traverse override	DB21, ...DBX12.5 DB21, ...DBX16.5 DB21, ...DBX20.5	DB3200.DBX1000.5 DB3200.DBX1004.5 DB3200.DBX1008.5
Plus traversing keys and minus traversing keys	DB21, ...DBX12.6/7 DB21, ...DBX16.6/7 DB21, ...DBX20.6/7	DB3200.DBX1000.6/7 DB3200.DBX1004.6/7 DB3200.DBX1008.6/7
Geometry axis machine function INC1 ... continuous	DB21, ...DBB13/17/21	DB3200.DBB1001/1005/1009

4.11.3.5 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
DRF selected	DB21, ...DBX24.3	DB1700.DBX0.3
Handwheel active (3, 2, 1)	DB21, ...DBX40.0-2 DB21, ...DBX46.0-2 DB21, ...DBX52.0-2	DB3300.DBX1000.0-1 DB3300.DBX1004.0-1 DB3300.DBX1008.0-1
Geometry axis travel request	DB21, ...DBX40.4/5 DB21, ...DBX46.4/5 DB21, ...DBX52.4/5	DB3300.DBX1000.4/5 DB3300.DBX1004.4/5 DB3300.DBX1008.4/5
Travel command plus and travel command minus	DB21, ...DBX40.6/7 DB21, ...DBX46.6/7 DB21, ...DBX52.6/7	DB3300.DBX1000.6/7 DB3300.DBX1004.6/7 DB3300.DBX1008.6/7
Geometry axis active, machine function INC1 ... continuous	DB21, ...DBB41/47/53	DB3300.DBB1001/1005/1009
Handwheel override active, for path axes	DB21, ...DBX33.3	DB3300.DBX1.3
Activate handwheel 1 as contour handwheel; activate handwheel 2 as contour handwheel; activate handwheel 3 as contour handwheel	DB21, ...DBX30.0 DB21, ...DBX30.1 DB21, ...DBX30.2	DB3200.DBX14.0 DB3200.DBX14.1
Contour-handwheel simulation ON	DB21, ...DBX30.3	DB3200.DBX14.3
Contour-handwheel-simulation negative direction	DB21, ...DBX30.4	DB3200.DBX14.4
Handwheel 1 active as contour handwheel; handwheel 2 active as contour handwheel; handwheel 3 active as contour handwheel	DB21, ...DBX37.0 DB21, ...DBX37.1 DB21, ...DBX37.2	DB3300.DBX5.0
Handwheel 1 active as contour handwheel; handwheel 2 active as contour handwheel; handwheel 3 active as contour handwheel	DB21, ...DBX100.5 DB21, ...DBX101.5 DB21, ...DBX102.5	DB3300.DBX5.1
Orientation axis travel request	DB21, ...DBX332.4/5 DB21, ...DBX336.4/5 DB21, ...DBX340.4/5	-

4.11.3.6 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate/spindle override	DB21, ...DBB0	DB380x.DBB0
Override active	DB21, ...DBX1.7	DB380x.DBX1.7
Axial delete distance-to-go	DB21, ...DBX2.2	DB380x.DBX2.2
Activate handwheel (1, 2, 3)	DB21, ...DBX4.0-2	DB380x.DBX4.0-1
Traversing key lock	DB21, ...DBX4.4	DB380x.DBX4.4
Rapid traverse override	DB21, ...DBX4.5	DB380x.DBX4.5
Plus traversing keys and minus traversing keys	DB21, ...DBX4.6/7	DB380x.DBX4.6/7
Machine function: 1 INC, 10 INC, 100 INC, 1000 INC, 10000 INC, Var. INC, continuous	DB21, ...DBX5.0-6	DB380x.DBX5.0-6
Activated fixed point travel in JOG (binary coded)	DB21, ...DBX13.0-2	DB380x.DBX1001.0-2

4.11.3.7 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Position reached with coarse/fine exact stop	DB21, ...DBX60.6/7	DB390x.DBX0.6/7
Handwheel active (1, 2, 3)	DB21, ...DBX64.0-2	DB390x.DBX4.0/1
Plus and minus axis/spindle travel request	DB21, ...DBX64.4/5	DB390x.DBX4.4/5
Travel command plus and travel command minus	DB21, ...DBX64.6/7	DB390x.DBX4.6/7
Active machine function INC1 ... continuous	DB21, ...DBB65	DB390x.DBB5
Handwheel override active, for positioning axes and concurrent positioning axes	DB21, ...DBX62.1	DB390x.DBX2.1
Fixed point travel in JOG active (binary coded)	DB21, ...DBX75.0-2	DB390x.DBX1001.0-2
Fixed point reached (binary coded)	DB21, ...DBX75.3-5	DB390x.DBX1001.3-5

K3: Compensation

5

5.1 Introduction

Accuracy errors

The accuracy of machine tools is impaired as a result of deviations from the ideal geometry, power transmission faults and measuring system errors. Temperature differences and mechanical forces often result in great reductions in precision when large workpieces are machined.

Compensation functions

Some of these deviations can usually be measured during commissioning and then compensated for during operation on the basis of values read by the positional actual-value encoder and other sensory devices. State-of-the-art CNC controls have compensation functions that are active on an axis for axis basis.

For SINUMERIK 840D sl, the following compensation functions are available:

- Temperature compensation
- Backlash compensation
- Interpolatory compensation
  - Compensation of leadscrew errors and measuring system errors
  - Compensation of sag and angularity errors
- Dynamic feedforward control (following error compensation)
- Friction compensation (quadrant error compensation)
  - Conventional friction compensation
  - Quadrant error compensation with neural networks
- Electronic counterweight

Parameterization

These compensation functions can be set for each machine individually with axis-specific machine data.

Activation

The compensations are active in all operating modes of the control as soon as the input data are available. Any compensations that require the position actual value are not activated until the axis reaches the reference point.

Position display

The normal actual-value and setpoint position displays ignore the compensation values and show the position values of an ideal machine. The compensation values are output in the "Service axes" display in the "Diagnosis" operating area.

5.2 Temperature compensation

5.2.1 Description of functions

Deformation due to temperature effects

Heat generated by the drive equipment or high ambient temperatures (e.g. caused by sunlight, drafts) cause the machine base and parts of the machinery to expand. This expansion depends, among other things, on the temperature and on the thermal conductivity of the machine parts.

Effects

Due to the thermal expansion of the machinery, the actual positions of the axes change depending on temperature. This has a negative impact on the precision of the workpieces being machined.

Temperature compensation

By activating the "temperature compensation" function, actual value changes due to temperature effects can be compensated on an axis-by-axis basis.

Sensor equipment

To provide effective temperature compensation, a number of temperature sensors for acquiring a temperature profile are needed in addition to the actual position data from existing encoders.

Since temperature-dependent changes occur relatively slowly, the PLC can acquire and preprocess the temperature profile in a minutes cycle, for example.

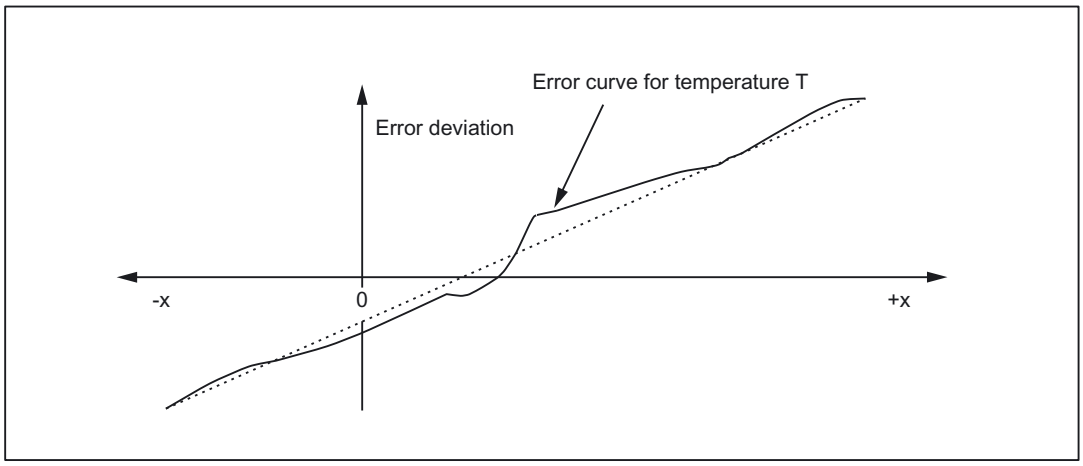
Error curves

In order to implement temperature compensation, the actual-value offsets over the positioning range of the axis must be measured at a given temperature (T) and plotted. This produces an error curve for this temperature value. Error curves must be produced for different temperatures.

Error curve characteristic

If an axis position reference point P_0 is selected, an offset in the reference point (corresponds to the "position-independent component" of the temperature compensation) can be observed as the temperature changes, and because of the change in length an additional offset in the other position points, which increases with the distance to the reference point (corresponds to the "position-dependent component" of the temperature compensation).

The error curve for a given temperature T can generally be represented with sufficient accuracy by a straight line with a temperature dependent gradient and reference position.



Compensation equation

The compensation value ΔK_x is calculated on the basis of current actual position P_x of this axis and temperature T according to the following equation:

$\Delta K_x = K_0 (T) + \tan\beta (T) * (P_x - P_0)$

The meaning is as follows:

ΔK_x:	Temperature compensation value of axis at position P_x
K_0:	Position-independent temperature compensation value of axis
P_x:	Actual position of axis
P_0	Reference position of axis
tanβ	Coefficient for the position-dependent temperature compensation (corresponds to the gradient of the approximated error line)

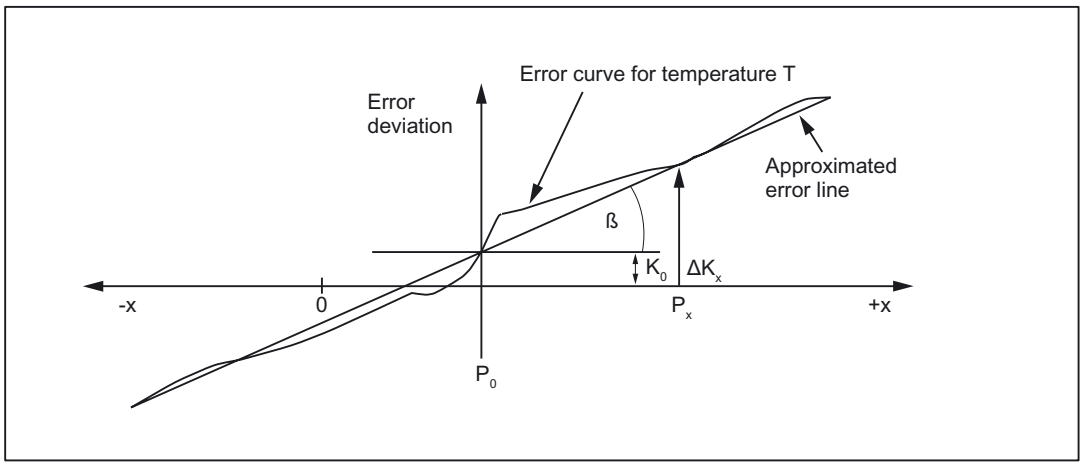


Figure 5-1 Approximated error line for temperature compensation

Activation

The following conditions must be fulfilled so that the temperature compensation can be activated:

1. The compensation type is selected (MD32750, see "Temperature compensation type and activation (Page 348)").
2. The parameters for the compensation type are defined (see "Temperature-dependent parameters (Page 348)").
3. The axis is referenced:

   DB31, ... DBX60.4 or 60.5 =1 (referenced/synchronized 1 or 2 respectivelly)

As soon as these conditions are fulfilled, the temperature compensation value for the position actual value is added to the setpoint in all modes and the machine axis traverses through this distance. If the compensation value ΔK_x is positive, the axis moves in the negative direction.

If the reference position is then lost, e.g. because the encoder frequency has been exceeded (DB31, ... DBX60.4 or 60.5 = 0), compensation processing is deactivated.

Clock cycle

The compensation values are determined in the interpolation cycle.

Display

The total compensation value calculated from the temperature and sag compensation functions belonging to the actual position is output in the "Service axes" display in the "Diagnosis" operating area.

Parameter adaptation for temperature changes

Since the approximated error line applies only to the instantaneous temperature value, the parameters of the error lines that are newly generated when the temperature rises or falls must be sent to the NCK again. Only in this way can expansion due to heat always be correctly compensated.

When temperature T changes, the parameters which are temperature-dependent, i.e. (K_0, tanβ and P_0) also change and can thus always be overwritten by the PLC or by means of a synchronized action.

It is thus possible for the machine-tool manufacturer to emulate the mathematical and technological relationship between the axis positions and temperature values via the PLC user program and thus calculate the various parameters for the temperature compensation. The temperature parameters are transferred to the NCK using the variable services (FB2 (GET) "Read data" and FB3 (PUT) "Write data").

For more information on handling and supplying FB2 and FB3 see:
Reference:
Function Manual Basic Functions; Basic PLC Program (P3)

Smooth the compensation value

To prevent overloading of the machine or tripping of monitoring functions in response to step changes in the temperature compensation parameters, the compensation values are distributed over several IPO cycles by an internal control function as soon as they exceed the maximum compensation value specified for each IPO cycle (MD32760, see "Maximum compensation value per IPO clock cycle (Page 349)").

5.2.2 Commissioning

5.2.2.1 Temperature-dependent parameters

SD43900, SD43910, SD43920

Error curves for different temperatures can be defined for each axis. For each error curve the following parameters must be determined and then entered in the setting data:

- Position-independent temperature compensation value K_0:

  SD43900 $SA_TEMP_COMP_ABS_VALUE
- Reference position P_0 for position-dependent temperature compensation:

  SD43920 $SA_TEMP_COMP_REF_POSITION
- Gradient tanβ for position-dependent temperature compensation:

  SD43910 $SA_TEMP_COMP_SLOPE

5.2.2.2 Temperature compensation type and activation

MD32750

The temperature compensation type is selected and the temperature compensation activated using the axis-specific machine data:

MD32750 $MA_TEMP_COMP_TYPE (temperature compensation type)

Bit	Value	Significance	Associated parameters
0	Position independent temperature compensation		SD43900
	0	Not active	
	1	Active	
1	Position-dependent temperature compensation		SD43920, SD43910
	0	Not active	
	1	Active	
2	Temperature compensation in tool direction		MD20390 $MC_TOOL_TEMP_COMP_ON (Activate temperature compensation tool length)
	0	Not active	
	1	Active	

5.2.2.3 Maximum compensation value per IPO clock cycle

MD32760

The maximum possible compensation value per IPO cycle, i.e. the maximum distance that can be traversed in an IPO cycle as a result of the temperature compensation, is limited using machine data:

MD32760 $MA_COMP_ADD_VELO_FACTOR (velocity increase as a result of compensation)

The specified value acts as a factor and is referred to the maximum axis velocity (MD32000 $MA_MAX_AX_VELO).

MD32760 also limits the maximum gradient of the error line (tanß) of the temperature compensation.

5.2.3 Example

5.2.3.1 Commissioning the temperature compensation for the Z axis of a lathe

Commissioning of temperature compensation is described below using the example of a Z axis on a lathe.

Determining the error characteristic of the Z axis

In order to determine the temperature-dependent error characteristic of the Z axis, proceed as follows:

- Uniform temperature increase by traversing the axis across the whole Z axis traversing range (in the example: from 500 mm to 1500 mm)
- Measuring the axis position in increments of 100 mm
- Measuring the actual temperature at the leadscrew
- Executing a traversing measuring cycle every 20 minutes

The mathematical and technological relationships and the resulting parameters for temperature compensation are derived from the recorded data. The calculated deviation errors for a specific temperature, which refer to the actual position of the Z axis displayed by the NC, are represented in graphic form in the diagram below.

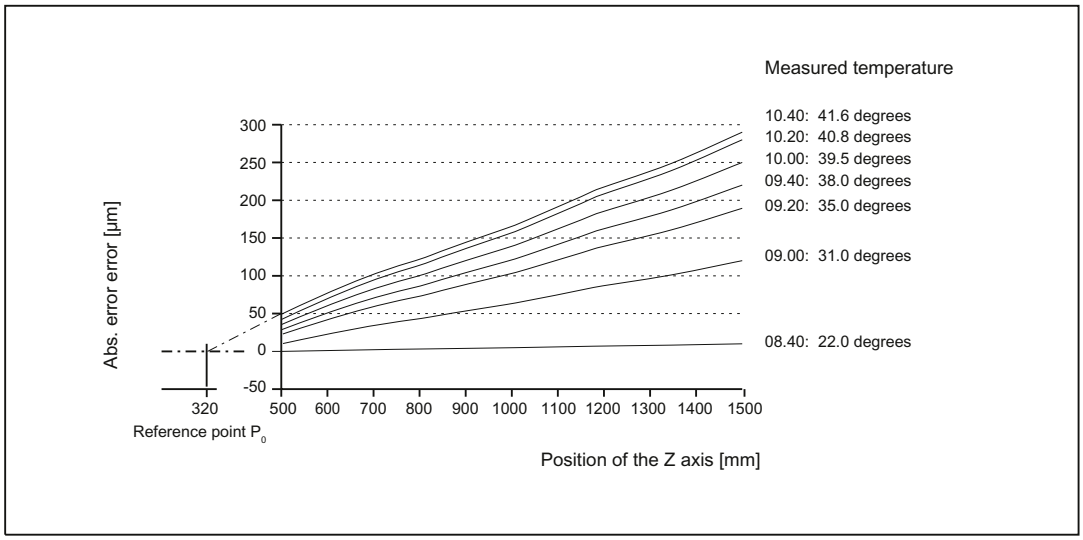


Figure 5-2 Error curves determined for the Z axis

Specifying parameters

The temperature compensation parameters must now be determined on the basis of the measurement results (see diagram above).

Reference position P_0

As the diagram above illustrates, there are basically two methods of parameterizing reference position P_0:

1. $P_0 = 0$ with position-independent temperature compensation value $K_0 \neq 0$
2. $P_0 \neq 0$ with position-independent temperature compensation value $K_0 = 0$

In this case, version 2 is chosen, which means that the position-independent temperature compensation value is always 0. The temperature compensation value therefore only consists of the position-dependent component.

The following parameters are obtained:

- MD32750 $MA_TEMP_COMP_TYPE = 2

  (only position-dependent temperature compensation active)
- P_0 = 320 mm → SD43920 $SA_TEMP_COMP_REF_POSITION = 320

  (reference position for position-dependent temperature compensation)

Coefficient tanβ (T)

In order to determine the dependency of coefficient tanβ of the position-dependent temperature compensation on the temperature, the error curve gradient is plotted against the measured temperature:

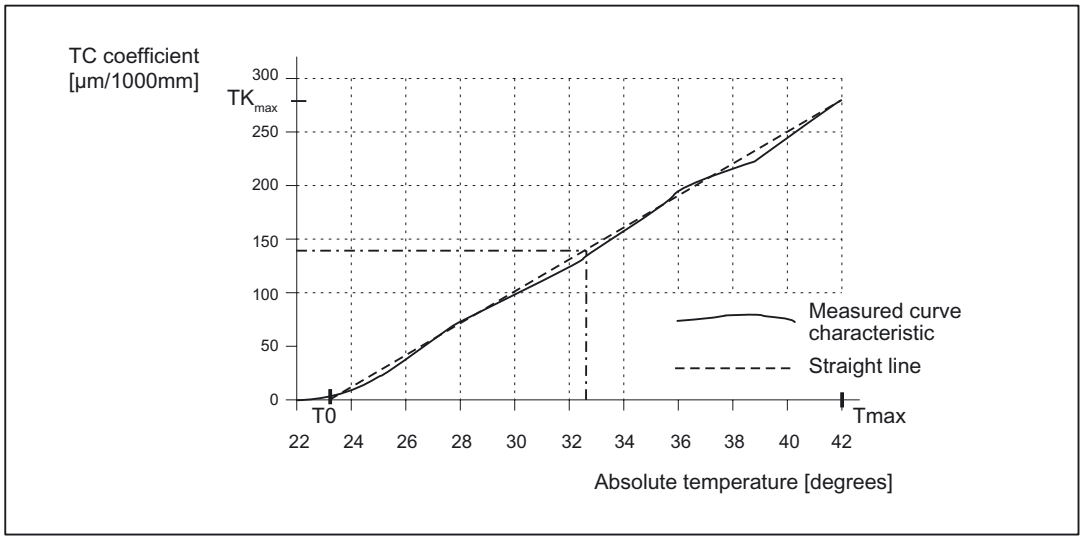


Figure 5-3 Characteristic of coefficient tanβ as a function of measured temperature T

With the appropriate linearization, coefficient tanβ depends on T as follows:

$\tan\beta(T) = (T - T_0) * TK_{max} * 10^{-6} / (T_{max} - T_0)$

with

T_0 = temperature at which position-dependent error = 0; [degrees]

T_{max} = maximum measured temperature; [degrees]

TK_{max} = temperature coefficient at T_{max}; [µm/1000 mm]

Therefore, based on the values from the above diagram:

T_0 = 23°

T_{max} = 42°

TK_{max} = 270 µm/1000 mm

and tanß (T) is therefore:

$\tan\beta(T)$ = (T - 23) [degrees] * 270 [µm/1000 mm] * 10^{-6} / (42 - 23) [degrees]
= (T - 23) [degrees] * 14.21 [µm/1000 mm] * 10^{-6}

Example:

At a temperature of T = 32.3 degrees, therefore: tanβ = 0.000132

PLC user program

The formula given above must be used in the PLC user program to calculate the coefficient tanβ (T) which corresponds to the measured temperature; this must then be written to the following NCK setting data:

SD43910 $SA_TEMP_COMP_SLOPE (gradient for position-dependent temperature compensation)

According to the example above:

SD43910 $SA_TEMP_COMP_SLOPE = 0.000132

5.3 Backlash compensation

5.3.1 Description of functions

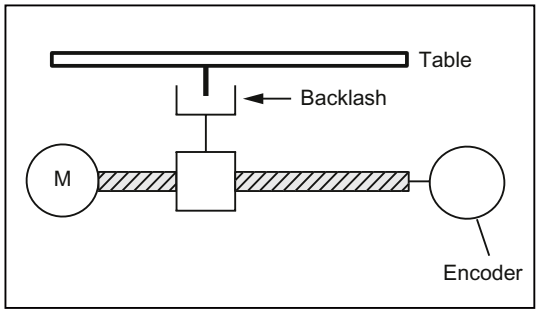
Mechanical backlash

When force is transmitted between a moving machine part and its drive (e.g. ball screw), there is normally a small amount of backlash because adjusting mechanical parts so that they are completely free of backlash would result in too much wear and tear on the machine. Thus, backlash (play) can occur between the machine component and the measuring system.

Effects

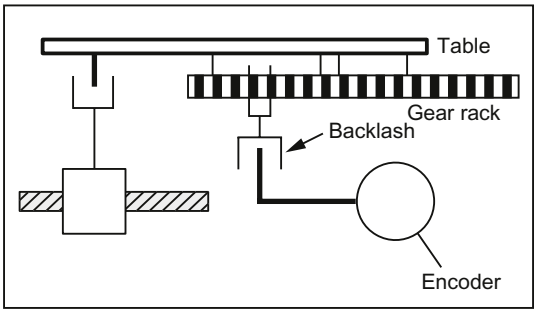
In the case of axes/spindles with indirect measuring systems, mechanical backlash falsifies the traversing path. For example, when the direction of movement is reversed, an axis will travel too much or too little by the amount of the backlash.

Positive backlash (normal case)



The encoder leads the machine part (e.g. table). Since the actual position acquired by the encoder also leads the real actual position of the table, the table travels too short a distance.

Negative backlash



The encoder lags behind the machine part (e.g. table). The table travels too far as the actual position of the table leads the actual position detected by the encoder.

Backlash compensation

To compensate for backlash, the axis-specific actual value is corrected by the axis-specific backlash amount specified when commissioning (MD32450, see "Backlash (Page 353)") every time the axis/spindle changes direction.

Activation

Backlash compensation is always active in all operating modes after reference point approach.

Display

The compensation value associated with the actual position is output as the total compensation calculated from "LEC" and "backlash compensation" in the "Service axes" display in the "Diagnosis" operating area.

5.3.2 Commissioning

5.3.2.1 Backlash

MD32450

The correction value for the backlash compensation is entered into the machine data for each axis/spindle:

MD32450 $MA_BACKLASH (backlash)

For positive backlash (normal case), the compensation value should be entered as a positive value and for a negative backlash, as negative value.

2nd measuring system

If there is a 2nd measuring system for the axis/spindle, a backlash compensation must be entered for this too. As the second measuring system is mounted in a different way from the first measuring system, the backlash can be different from that of the first measuring system.

When the measuring system is switched over the associated compensation value is always automatically activated.

5.3.2.2 Weighting factor for backlash

MD32452

The backlash can be weighted by a factor dependent on the particular parameter set.

This weighting factor is set to between 0.01 and 100.0 in the following machine data:

MD32452 $MA_BACKLASH_FACTOR (backlash weighting factor)

The factory default setting is 1.0.

Application: e.g. compensation of gear-stage-dependent backlash.

5.3.2.3 Applying the backlash compensation step-by-step

MD36500

The user has the option of applying the backlash compensation value gradually in several increments when the relevant axis reverses direction. This prevents a setpoint step change on the axes from causing corresponding errors.

The contents of the following axis-specific machine data determine the increment with which the backlash compensation value (MD32450) is applied.

MD36500 $MA_ENC_CHANGE_TOL (max. tolerance on position actual value switchover)

Please note that the backlash compensation is only fully calculated after <n> servo cycles (<n> = MD32450 / MD36500). An excessive time span can cause the triggering of a zero speed monitoring alarm.

If MD36500 > MD32450, then the compensation is performed in one servo cycle.

5.4 Interpolatory compensation

5.4.1 General properties

Function

The "Interpolatory compensation" function allows position-related dimensional deviations (for example, leadscrew and measuring system errors, sag and angularity errors) to be corrected.

The compensation values are measured during commissioning and stored in a table as a position-related value. During operation, the corresponding axis is compensated between interpolation points during linear interpolation.

Methods

Within "interpolatory compensation", a distinction is made between the two following compensation methods:

- Compensation of leadscrew errors and measuring system errors
- Compensation of sag and angularity errors

Many of the characteristics of these two compensation methods are identical and are therefore subsequently described for the two methods.

Terminology

Important terms for "interpolatory compensation" are:

- Compensation value

  The difference between the axis position measured by the position actual-value encoder and the required programmed axis position (= axis position of the ideal machine). The compensation value is often also referred to as the correction value.

- Basic axis

  Axis whose setpoint position or actual position forms the basis for calculating a compensation value.

- Compensation axis

  Axis whose setpoint position or actual position is modified by a compensation value.

- Interpolation point

  A position of the base axis and the corresponding compensation value of the compensation axis.

- Compensation table

  Table of interpolation points and compensation values (see below)

- Compensation relation

  Assignment of the base axis and the corresponding compensation axis and the reference to the corresponding compensation table.

Compensation tables

Because the mentioned dimensional deviations directly affect the accuracy of workpiece machining, they must be compensated for by the relevant position-dependent compensation values. The compensation values are derived from measured error curves and entered in the control in the form of compensation tables during commissioning. A separate table must be created for each compensation relation.

Entering compensation tables

The size of the compensation table, i.e. the number of interpolation points, must first be defined in a machine data. After the next POWER ON, the compensation tables are generated by the NC and preassigned a value of "0".

The compensation values and additional table parameters are entered in the compensation tables using special system variables. Data can be loaded in two different ways:

- By starting an NC program with the parameter values.
- By transferring the compensation tables from a PC via the serial interface of the PCU.

Note

Compensation tables can be output from the "Services" operating area via the serial interface on the PCU and loaded back after editing.

Note

Compensation tables can only be loaded if:

MD32700 $MA_ENC_COMP_ENABLE (interpolatory compensation) = 0

or

MD32710 $MA_CEC_ENABLE (enable sag compensation) = 0

These compensation values are not lost when the control is switched off because they are stored in the static user memory. They can be updated if necessary (e.g. following re-measuring because of machine aging).

⚠ CAUTION

When the setting in machine data:

- MD18342 $MN_MM_CEC_MAX_POINTS[<t>] (maximum number of interpolation points for sag compensation); <t> = Index of the compensation table

or

- MD38000 $MA_MM_ENC_COMP_MAX_POINTS (number of interpolation points for interpolatory compensation)

the static user memory is automatically re-allocated when the system boots.

All the user data in the static NC memory (e.g. drive and HMI machine data, tool offsets, part programs, compensation tables, etc.) are deleted.

References:
Function Manual, Extended Functions; Memory Configuration (S7)

Logging

Compensation tables are not saved with the series start-up file.

To archive compensation tables, they must be output via the serial interface on the PCU. A distinction is made between the following compensation types in the operating area "Services" > "Data out":

- LEC/measuring system error compensation (%_N_AX_EEC_INI)
- Sag/angularity compensation (%_N_AX_CEC_INI)
- Quadrant error compensation (%_N_AX_QEC_INI)

Compensation tables can also be saved as an archive file with HMI Advanced.

Linear interpolation between interpolation points

The traversing path to be compensated delineated by the start and end positions is divided up into several (number depends on error curve shape) path segments of equal size (see diagram below). The actual positions that limit these sub-paths are referred to as "interpolation points" below. A compensation value must be entered for each interpolation point (actual position) during commissioning. The compensation value applied between 2 interpolation points is generated on the basis of **linear interpolation** using the compensation values for the adjacent interpolation points (i.e. adjacent interpolation points are linked along a line).

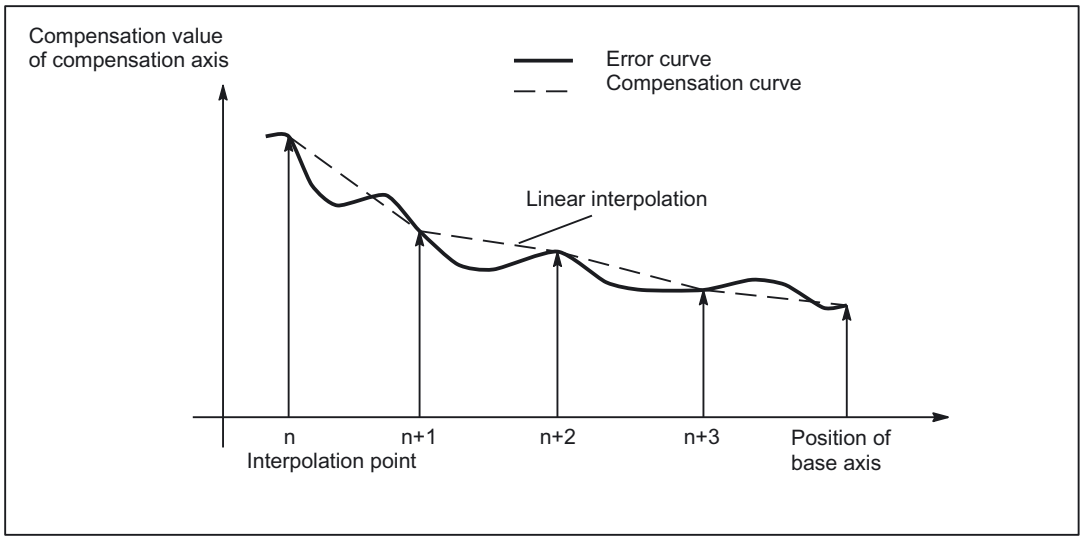


Figure 5-4 Linear interpolation between the interpolation points

Compensation value at reference point

The compensation table should be structured such that the compensation value at the reference point is "zero".

5.4.2 Compensation of leadscrew errors and measuring system errors

5.4.2.1 Description of functions

Leadscrew and measuring system errors

The measuring principle of "indirect measurement" on NC-controlled machines is based on the assumption that the lead of the ball screw is constant at any point within the traversing range, so that the actual position of the axis can be derived from the position of the drive spindle (ideal case). However, manufacturing tolerances result in dimensional deviations of varying degrees of severity on spindles (so-called leadscrew errors).

Added to this are the dimensional deviations (differences in reference division) caused by the measuring system as well as its mounting on the machine (so-called measuring system errors), plus any machine-dependent error sources.

Compensation

With "measuring system error compensation" (referred to below as **MSEC**), the base and compensation axes are always identical. It is therefore an **axial compensation** for which a definition of the base axis and compensation axis in the compensation table is not necessary.

Note

The leadscrew error compensation function is part of the measuring system error compensation system.

The principle of the MSEC is to modify the axis-specific position actual value by the assigned compensation value in the interpolation cycle and to apply this value to the machine axis for immediate traversal. A positive compensation value causes the corresponding machine axis to move in the negative direction.

The magnitude of the compensation value is not limited and is not monitored. In order to avoid impermissibly high velocities and accelerations caused by compensation, small compensation values must be selected. Large compensation values can cause other axis monitoring functions to output alarms (e.g. contour monitoring, speed setpoint limitation).

If the axis to be compensated has a 2nd position measuring system, a separate compensation table must be created and activated for each measuring system. The correct table is automatically used when switching between measuring systems.

Preconditions/activation

The MSEC is only active until the following pre-conditions:

- The compensation values are stored in the static user memory and are active (after POWER ON).
- The function has been activated for the relevant machine axis:

  MD32700 $MA_ENC_COMP_ENABLE [<e>] = 1

  with: <e> = Position measuring system
  <e> = **0** Measuring system **1**
  <e> = **1** Measuring system **2**

- The axis has been referenced:

  DB31, ... DBX60.4 or 60.5 =1 (referenced/synchronized 1 or 2)

As soon as these conditions have been fulfilled, the axis-specific actual value is modified by the compensation value in all modes and traversed by the machine axis immediately.

If the reference is then lost, e.g. because the encoder frequency has been exceeded (DB31, ... DBX60.4 or 60.5 = 0), compensation processing is deactivated.

5.4.2.2 Commissioning

Number of compensation interpolation points (MD38000)

For every machine axis and for every measuring system (if a 2nd measuring system is installed), the number of reserved interpolation points of the compensation table must be defined and the necessary memory reserved with the following machine data:

MD38000 $MA_MM_ENC_COMP_MAX_POINTS[<e>,<AXi>]

with: <e> = Position measuring system
<e> = 0 Measuring system 1
<e> = 1 Measuring system 2

<AXi> = Axis name (e.g. X1, Y1, Z1)

The required number of compensation interpolation points is calculated as follows:

$$\text{MM_ENC_COMP_MAX_POINTS [e,AXi]} = \frac{\$AA_ENC_COMP_MAX[e,AXi] - \$AA_ENC_COMP_MIN[e,AXi]}{\$AA_ENC_COMP_STEP[e,AXi]} + 1$$

Measuring system-specific parameters of the compensation table

The position-related compensations as well as additional table parameters should be saved in the form of system variables for each machine axis as well as for each measuring system (if a 2nd measuring system is being used):

- **$AA_ENC_COMP[<e>,<N>,<AXi>]**

  (Compensation value for interpolation point N in the compensation table)

  <N> = interpolation point (axis position)

  For every individual interpolation point the compensation value must be entered in the table.

  <N> is limited by the maximum number of interpolation points of the particular compensation table (MD38000 $MA_MM_ENC_COMP_MAX_POINTS):

  0 ≤ N ≤ MD38000 -1

  The size of the compensation value is not limited.

  Note

  The first and last compensation values remain active over the entire traversing range; i.e. these values should be set to "0" if the compensation table does not cover the entire traversing range.

- **$AA_ENC_COMP_STEP[<e>,<AXi>] (distance between interpolation points)**

  The distance between interpolation points defines the distance between the compensation values in the relevant compensation table.

- **$AA_ENC_COMP_MIN[<e>,<AXi>] (initial position)**

  The initial position is the axis position at which the compensation table for the relevant axis begins (≙ interpolation point 0).

  The compensation value for the initial position is $AA_ENC_COMP[<e>,**0**,<AXi>)].

  The compensation value of interpolation point 0 is used for all positions smaller than the initial position (does not apply for tables with modulo function).

- **$AA_ENC_COMP_MAX[<e>,<AXi>] (end position)**

  The end position is the axis position at which the compensation table for the relevant axis ends (≙ interpolation point <k>).

  The compensation value for the end position is $AA_ENC_COMP[<e>,**<k>**,<AXi>)].

  The compensation value of interpolation point <k> is used for all positions larger than the end position (exception for table with modulo function).

  The following supplementary conditions apply to interpolation point <k>:

  – for k = MD38000 - 1:

    The compensation table is fully utilized!

  – for k < MD38000 - 1:

    The compensation table is not fully utilized. Compensation values entered in the table that are greater than k are ignored.

  – for k > MD38000 - 1:

    The compensation table is limited by a control function which reduces the end position. Compensation values that are greater than k are ignored.

- **$AA_ENC_COMP_IS_MODULO[<e>,<AXi>] (compensation with modulo function)**

  System variable to activate/deactivate the compensation with modulo function:

  – $AA_ENC_COMP_IS_MODULO[<e>,<AXi>] **= 0**: Compensation **without** modulo function

  – $AA_ENC_COMP_IS_MODULO[<e>,<AXi>] **= 1**: Compensation **with** modulo function

  When compensation with modulo function is activated, the compensation table is repeated cyclically, i.e. the compensation value at position $AA_ENC_COMP_MAX (≙ interpolation point $AA_ENC_COMP[<e>,<k>,<AXi>]) is immediately followed by the compensation value at position $AA_ENC_COMP_MIN (≙ interpolation point $AA_ENC_COMP[<e>,<0>,<AXi>]).

  For rotary axes with modulo 360° degrees it is therefore suitable to program 0° ($AA_ENC_COMP_MIN) as the initial position and 360° ($AA_ENC_COMP_MAX) as the end position.

  The compensation values entered for these two positions should be the same as otherwise the compensation value jumps from MAX to MIN at the transition point and vice versa.

  ⚠ CAUTION

  When the compensation values are entered, it is important that all interpolation points within the defined range are assigned a compensation value (i.e. there should be no gaps). Otherwise, the compensation value that was left over from previous entries at these positions is used for these interpolation points.

Note

Table parameters containing position information are automatically converted when the system of units is changed (change from MD10240 $MN_SCALING_SYSTEM_IS_METRIC).

The position information is always interpreted in the actual system of units. Conversion must be implemented externally.

Automatic conversion of the position data can be configured as follows:

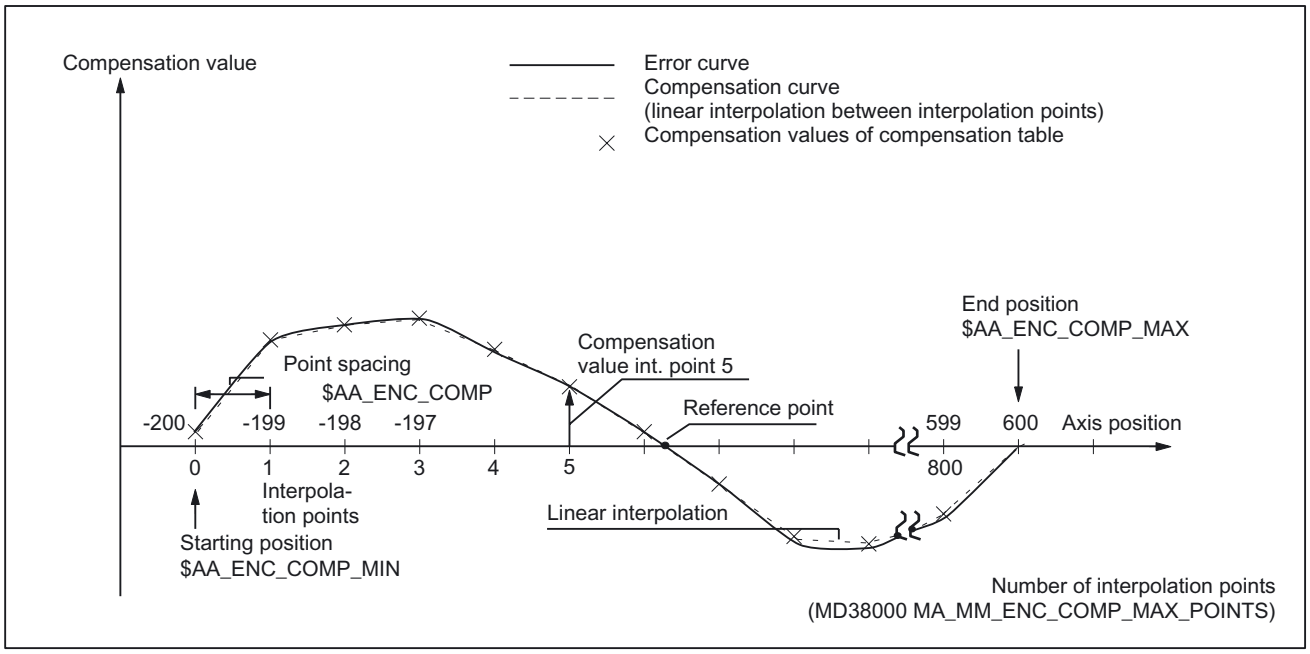
MD10260 $MN_CONVERT_SCALING_SYSTEM = 1

External conversion is no longer necessary.

Reference:
Function Manual Basic Functions; Velocities, Setpoint/Actual Value Systems, Closed-Loop Control (G2)

5.4.2.3 Example

The following example shows compensation value inputs for machine axis X1.




```
Program code                          Comment
%_N_AX_EEC_INI
CHANDATA(1)
$AA_ENC_COMP[0,0,X1]=0.0          ;   1st compensation value (interpolation point 0)
                                  ;   +0µm
$AA_ENC_COMP[0,1,X1]=0.01         ;   2nd compensation value (interpolation point 1)
                                  ;   +10µm
```

```
Program code                          Comment
$AA_ENC_COMP[0,2,X1]=0.012        ;   3rd compensation value (interpolation point 2)
                                  ;   +12µm
$AA_ENC_COMP[0,800,X1]=-0.0       ;   Last compensation value (interpolation point
                                      800)
$AA_ENC_COMP_STEP[0,X1]=1.0       ;   Distance between interpolation points 1.0 mm
$AA_ENC_COMP_MIN[0,X1]=-200.0     ;   Compensation starts at -200.0 mm
$AA_ENC_COMP_MAX[0,X1]=600.0      ;   Compensation ends at +600.0 mm
$AA_ENC_COMP_IS_MODULO[0,X1]=0    ;   Compensation without modulo function
M17
```

For this example, the configured number of interpolation points must be ≥ 801:

MD38000 $MM_ENC_COMP_MAX_POINTS ≥ 801

The memory required in the static user memory is 6.4 kbytes (8 bytes per compensation value).

5.4.3 Compensation of sag and angularity errors

5.4.3.1 Description of functions

Sag errors

Weight can result in position-dependent displacement and inclination of moved parts since it can cause machine parts and their guides to sag.

Also large workpieces (e.g. cylinders) sag under their own weight.

Angularity errors

If moving axes are not positioned in exactly the required angle (e.g. perpendicular) with respect to one another, increasingly serious positioning errors will occur as the deviation from zero point becomes greater.

Compensation

In contrast to the MSEC, the base and compensation axes need not be identical for "Sag compensation" or "Angularity error compensation", requiring an axis assignment in every compensation table.

In order to compensate for sag of one axis (base axis) which results from its own weight, the absolute position of another axis (compensation axis) must be influenced. "Sag compensation" is therefore an **inter-axis compensation**.

As illustrated in the diagram below, the further the machining head moves in the negative Y1 axis direction, the more the cross-arm sags in the negative Z1 axis direction.

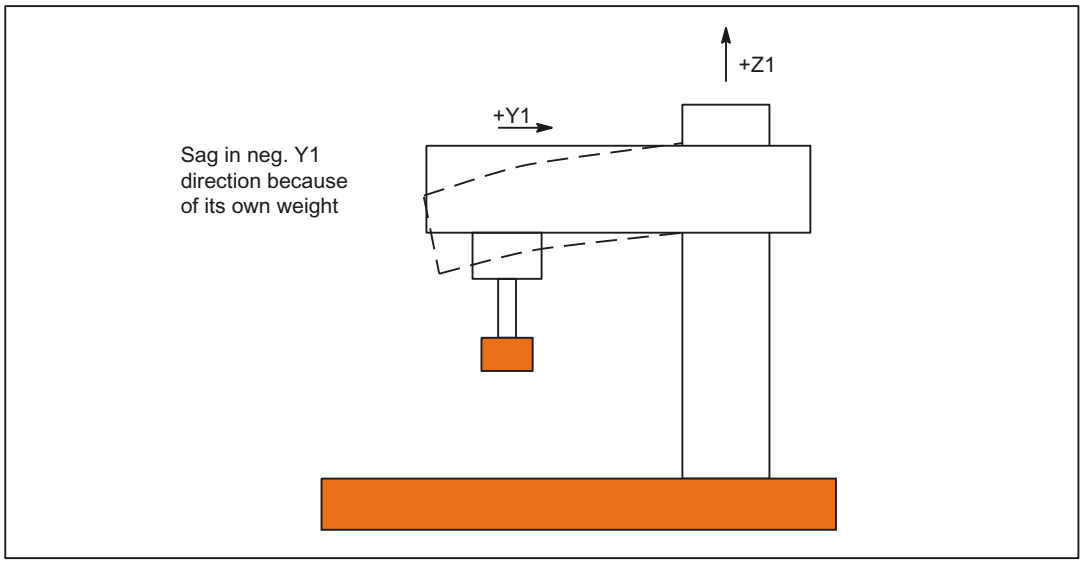


Figure 5-5 Example of sag caused by own weight

The error must be recorded in the form of a compensation table that contains a compensation value for the Z1 axis for every actual value position in the Y1 axis. It is sufficient to enter the compensation values for the interpolation points.

When the Y1 axis traverses, the control calculates the corresponding compensation value in the Z1 axis in interpolation cycles performing linear interpolation for positions between the interpolation points. This compensation is sent to the position control loop as an additional setpoint. A positive compensation value causes the corresponding machine axis to move in the negative direction.

Depending on the requirement, several compensation relations can be defined for one axis. The total compensation value results from the sum of all the compensation values of this axis.

Setting options

The many ways in which the compensation value for sag compensation can be produced/influenced are listed below (see diagram below).

1. An axis can be defined as the input variable (base axis) for **several** compensation tables (settable via system variables).
2. An axis can be defined as the recipient of the output variable (compensation axis) of **several** compensation tables (settable via system variable). The total compensation value is derived from the sum of the individual compensation values.

   The following definitions apply for the maximum number of possible compensation tables:

   – Maximum number of tables available for all axes:

     2 * maximum number of axes of system

   – Maximum number of tables configured for one particular compensation axis:

     1 * maximum number of axes of system
3. An axis can be both a base axis and a compensation axis at any one time. The programmed (required) position setpoint is always used to calculate the compensation values.
4. The scope of action of the compensation (starting and end position of the base axis) and the distance between the interpolation points can be defined for every compensation table (settable via system variables).
5. Compensation can be direction-dependent (settable via system variables).
6. Every compensation table has a modulo function for cyclic evaluation (settable via system variables).
7. A weighting factor by which the table value is multiplied (definable as a setting data which can therefore be altered by the part program, PLC or the user at any time) can be introduced for every compensation table.
8. Every compensation table can be multiplied with any other compensation table in pairs (i.e. also with itself) using the "table multiplication" function. A system variable is used to link the multiplication. The product is added to the total compensation value of the compensation axis.
9. The following options are available to activate the compensation:

   – Machine data:

     MD32710 $MA_CEC_ENABLE[<AXi>] (enable sag compensation)

     enables the sum of all compensation relationships for machine axis <AXi>.

   – With the setting data:

     SD41300 $SN_CEC_TABLE_ENABLE[<t>] (pre-assignment for the compensation table)

     evaluation of the compensation table [<t>] is enabled.

     It is thus possible e.g. to alter the compensation relationships either from the part program or from the PLC user program (e.g. switching over the tables), depending on the machining requirements.

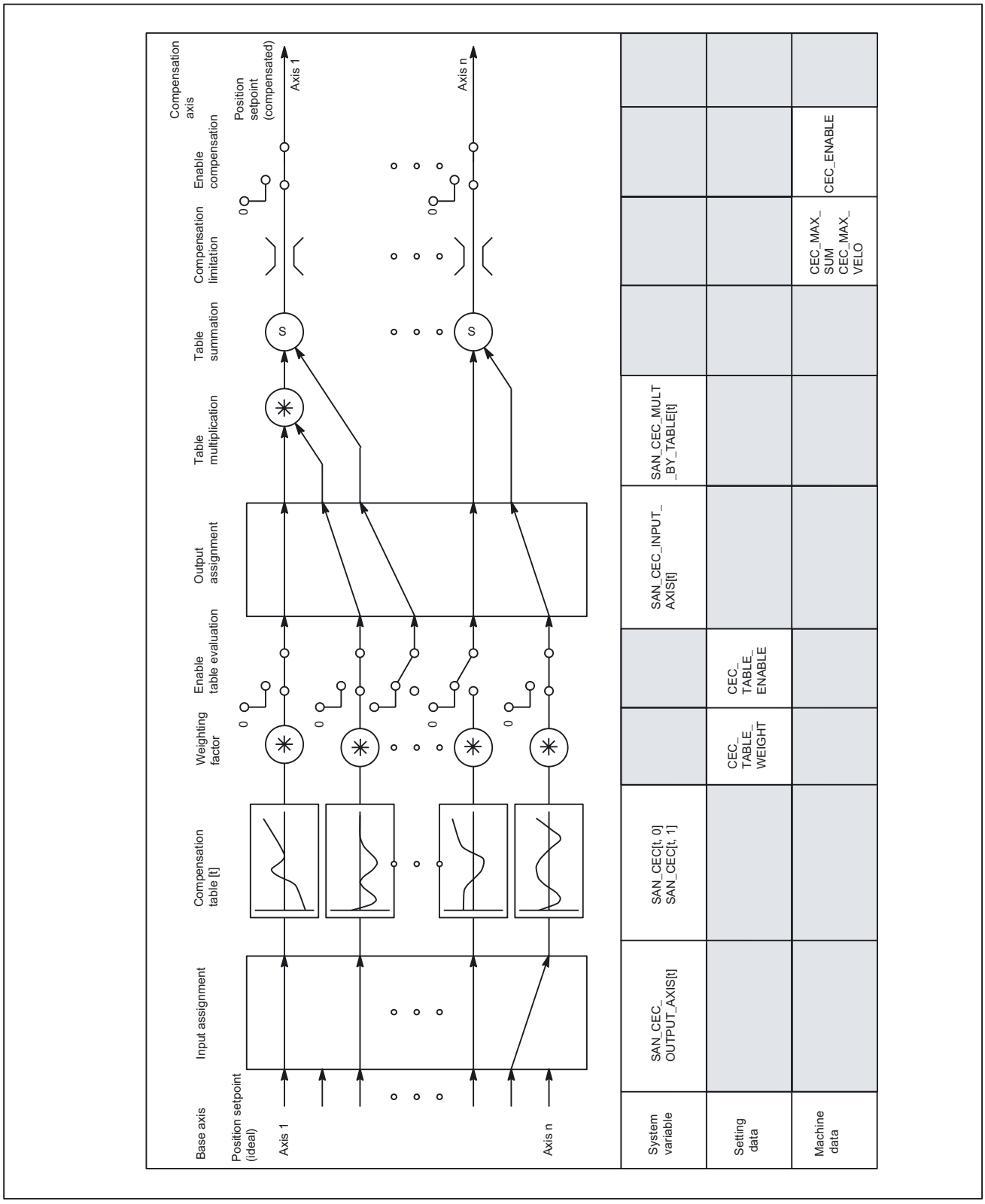


Figure 5-6 Generation of compensation value for sag compensation

Complex compensation

Since it is possible to use the position of an axis as the input quantity (base axis) for several tables, to derive the total compensation value of an axis from several compensation relationships (tables) and to multiply tables, it is also possible to implement sophisticated and complex beam sag and angularity error compensation systems.

This function also makes it possible to deal with different error sources efficiently. For example, it is possible to combine a table with a modulo function for a periodic recurring error component with a second table without a modulo function for an aperiodic error component for the same axis.

Leadscrew errors can also be compensated with this function by parameterizing an identical axis for the base and compensation axes. However, in contrast to the MSEC, measuring-system switchovers are not automatically registered in this case.

Preconditions/activation

The "sag compensation" function does not become active until the following conditions are fulfilled:

- The option "Sag compensation, multidimensional" is enabled.
- The function has been activated for the relevant machine axis (compensation axis):

  MD32710 $MA_CEC_ENABLE[<AXi>] = 1
- The compensation values are stored in the static user memory and are active (after POWER ON).
- Evaluation of the relevant compensation table has been enabled:

  SD41300 $SN_CEC_TABLE _ENABLE[<t>] = 1
- The current measuring system of the base and compensation axes has been referenced:

  DB31, ... DBX60.4 or 60.5 =1 (referenced/synchronized 1 or 2)

As soon as these conditions have been fulfilled the setpoint position of the compensation axis is altered in all modes with reference to the setpoint position of the base axis and the corresponding compensation value and is then immediately traversed by the machine axis.

If the reference is then lost, e.g. because the encoder frequency has been exceeded (DB31, ... DBX60.4 or 60.5 = 0), compensation processing is deactivated.

5.4.3.2 Commissioning

Number of compensation interpolation points (MD18342)

The number of required interpolation points in the compensation table must be defined for every compensation relationship and the memory space required is reserved using the following machine data:

MD18342 $MN_MM_CEC_MAX_POINTS[<t>] (maximum number of interpolation points for sag compensation)

with: <t> = Index of the compensation table
Permissible range: 0 ≤ t < 2 * maximum number of axes
t = 0: 1st compensation table
t = 1: 2nd compensation table
…

The required number of compensation interpolation points is calculated as follows:

$$MM_CEC_MAX_POINTS\ [t] = \frac{\$AN_CEC_MAX\ [t] - \$AN_CEC_MIN\ [t]}{\$AN_CEC_STEP\ [t]} + 1$$

Table parameters

The position-related compensation values as well as additional table parameters should be saved for every compensation relationship in the form of system variables:

- **$AN_CEC[<t>,<N>] (compensation value for interpolation point <N> of compensation table [<t>])**

  <N> = interpolation point (position of the basic axis)

  The particular compensation value of the compensation axis must be entered in the table for each individual interpolation point.

  <N> is limited by the maximum number of interpolation points in the relevant compensation table (MD18342 $MN_MM_CEC_MAX_POINTS):

  0 ≤ N ≤ MD18342 - 1

- **$AN_CEC_INPUT_AXIS[<t>] (basic axis)**

  Name of machine axis whose setpoint is to be used as the input for the compensation table [<t>].

- **$AN_CEC_OUTPUT_AXIS[<t>] (compensation axis)**

  Name of machine axis to which the output of the compensation table [<t>] is applied.

  Note

  In multi-channel systems the "general axis identifiers" AX1 ... must be defined, if the identifiers of machine axis and channel axis are identical.

- **$AN_CEC_STEP[<t>] (interpolation point distance)**

  The interpolation point distance defines the distance between the input values for the compensation table [<t>].

- **$AN_CEC_MIN[<t>] (initial position)**

  The initial position is the base axis position at which the compensation table [<t>] begins (≙ interpolation point 0).

  The compensation value for the initial position is $AN_CEC [<t>,0].

  The compensation value of interpolation point 0 is used for all positions smaller than the initial position (exception: table with modulo functions).

- **$AN_CEC_MAX[<t>] (end position)**

  The end position is the base axis position at which the compensation table [<t>] ends (≙ interpolation point <k>).

  The compensation value for the end position is $AN_CEC [<t>,**<k>**].

  The compensation value of interpolation point <k> is used for all positions larger than the end position (exception: table with modulo functions).

  The following supplementary conditions apply to interpolation point k:

  – for k = MD18342 - 1:

    The compensation table is fully utilized!

  – for k < MD18342 - 1:

    The compensation table is not fully utilized. The compensation values entered that are greater than k are ignored.

  – for k > MD18342 – 1:

    The compensation table is limited by a control function which reduces the end position. Compensation values that are greater than k are ignored.

- **$AN_CEC_DIRECTION[<t>] (direction-dependent compensation)**

  This system variable can be used to define whether the compensation table [<t>] should apply to both traversing directions of the base axis or only either the positive or negative direction:

  – $AN_CEC_DIRECTION[<t>] = 0:

    Table applies to both directions of travel of the base axis

  – $AN_CEC_DIRECTION[<t>] = 1:

    Table applies only to the positive traversing direction of the base axis

  – $AN_CEC_DIRECTION[<t>] = -1:

    Table applies only to the negative traversing direction of the base axis

  Possible applications:

  Position-dependent backlash compensation can be implemented using two tables, one of which affects the positive traversing direction, the other of which affects the negative traversing direction of the same axis.

- **$AN_CEC_MULT_BY_TABLE [<t>] (table multiplication)**

  With the table multiplication function, the compensation values of any table can be multiplied by those of any other table (or even by the same table). The product is added as an additional compensation value to the total compensation value of the compensation table.

  Syntax:

  $AN_CEC_MULT_BY_TABLE[<$t_1$>] = <$t_2$>

  with:

  <t_1>= Index of table 1 of compensation axis

  <t_2>= Number of table 2 of compensation axis

  Please note that the number and index of the same table are different! In general: Table number = table index + 1

- **$AN_CEC_IS_MODULO[<t>] (compensation with modulo function)**

  System variable to activate/deactivate the compensation with modulo function:

  – $AA_CEC_COMP_IS_MODULO[<t>] = **0**: Compensation **without** modulo function

  – $AA_CEC_COMP_IS_MODULO[<t>] = **1**: Compensation **with** modulo function

  When compensation with modulo function is activated, the compensation table is repeated cyclically, i.e. the compensation value at position $AN_CEC_MAX[<t>] (interpolation point $AN_CEC[<t>,<k>]) is immediately followed by the compensation value at position $AN_CEC_MIN[<t>] (interpolation point $AN_CEC[<t>,0]).

  These two compensation values should be the same as otherwise the compensation value jumps from MAX to MIN at the transition point and vice versa.

  If modulo compensation is to be implemented with a modulo rotary axis as base axis, the compensation table used has to be modulo calculated as well.

  Example:

  MD30300 $MA_IS_ROT_AX[AX1]=1 ; rotary axis

  MD30310 $MA_ROT_IS_MODULO[AX1]=1 ; modulo 360°

  $AN_CEC_INPUT_AXIS[0]=AX1

  $AN_CEC_MIN[0]=0.0

  $AN_CEC_MAX[0]=360.0

  $AN_CEC_IS_MODULO[0]=1

System of units

Table parameters containing position information are automatically converted when the system of units is changed (change from MD10240 $MN_SCALING_SYSTEM_IS_METRIC).

The position information is always interpreted in the actual system of units. Conversion must be implemented externally.

Automatic conversion of the position data can be configured as follows:

MD10260 $MN_CONVERT_SCALING_SYSTEM = 1

With this setting, the following axial machine data are activated:

MD32711 $MA_CEC_SCALING_SYSTEM_METRIC (system of units for sag compensation)

The measuring system for all tables effective for this axis is set in this machine data. Hereby all position entries are interpreted together with the calculated total compensation value in the configured measuring system. External conversions of position information are no longer necessary with a measuring system change.

Monitoring

To avoid excessive velocities and acceleration rates on the machine axis as a result of applying sag compensation, the total compensation value is monitored and limited to a maximum value.

The maximum possible total compensation value for sag compensation is defined on an axis-for-axis basis using the machine data:

MD32720 $MA_CEC_MAX_SUM (maximum compensation value for sag compensation)

If the determined total compensation value is greater than the maximum value, then a corresponding alarm is output. Program processing is not interrupted. The compensation value output as an additional setpoint is limited to the maximum value.

Further, changing the total compensation value is also axially limited:

MD32730 $MA_CEC_MAX_VELO (velocity change for sag compensation)

The specified value acts as a factor and is referred to the maximum axis velocity (MD32000 $MA_MAX_AX_VELO).

An appropriate alarm is signaled when the limit value is exceeded. Program processing is not interrupted. The path not covered because of the limitation is made up as soon as the compensation value is no longer subject to limitation.

5.4.3.3 Examples

Compensation table for sag compensation of the Y1 axis

The following example shows the compensation table for sag compensation of axis Y1. Depending on the position of the Y1 axis, a compensation value is applied to the Z1 axis. The 1st compensation table (<t> = 0) is used for this.

```
Program code                    Comment
%_N_NC_CEC_INI
CHANDATA(1)
$AN_CEC[0,0]=0                  ; 1st compensation value (interpolation point 0)
                                ; for Z1: ±0µm
$AN_CEC[0,1]=0.01               ; 2nd compensation value (interpolation point 1)
                                ; for Z1: +10µm
$AN_CEC[0,2]=0.012              ; 3rd compensation value (interpolation point 2)
                                ; for Z1: +12µm
...
```

```
Program code                          Comment
$AN_CEC[0,100]=0                      ; Last compensation value (interpolation point 101)
                                      ; for Z1: ±0µm
$AN_CEC_INPUT_AXIS[0]=(AX2)           ; Basic axis Y1
$AN_CEC_OUTPUT_AXIS[0]=(AX3)          ; Compensation axis Z1
$AN_CEC_STEP[0]=8                     ; Distance between interpolation points 8.0mm
$AN_CEC_MIN[0]=-400.0                 ; Compensation starts at Y1=-400mm
$AN_CEC_MAX[0]=400.0                  ; Compensation ends at Y1=+400mm
$AN_CEC_DIRECTION[0]=0                ; Table applies for both directions of travel of Y1.
$AN_CEC_MULT_BY_TABLE[0]=0
$AN_CEC_IS_MODULO[0]=0                ; Compensation without modulo function
M17                                   ;
```

For this example, the configured number of interpolation points must be ≥ 101:

MD18342 $MN_MM_CEC_MAX_POINTS [0] ≥ 101

The memory required in the static user memory is at least 808 bytes (8 bytes per compensation value).

Application for table multiplication

The following example for the compensation of machine foundation sagging illustrates an application of table multiplication.

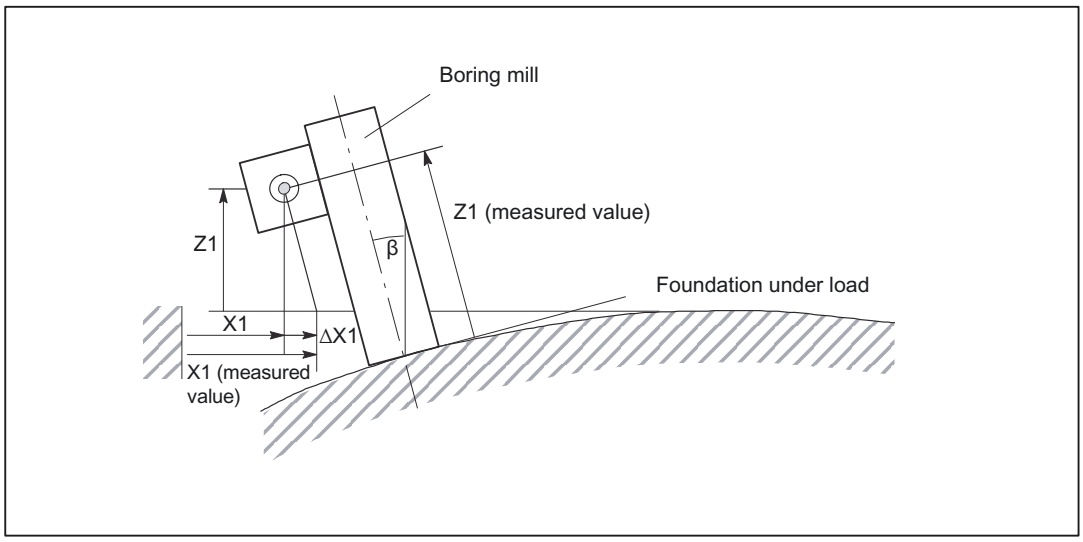


Figure 5-7 Compensation of sag in a foundation

On large machines, sagging of the foundation can cause inclination of the whole machine. For the boring mill shown in the diagram, for example, it is determined that compensation of the X1 axis is dependent both on the position of the X1 axis itself (since this determines angle of inclination β) and on the height of the boring mill (i.e. the position of the Z1 axis).

To implement compensation, the compensation values of the X1 and Z1 axes must be multiplied according to the following equation:

$\Delta X1 = Z1 * \sin\beta(X1) \approx Z1 * \beta(X1)$

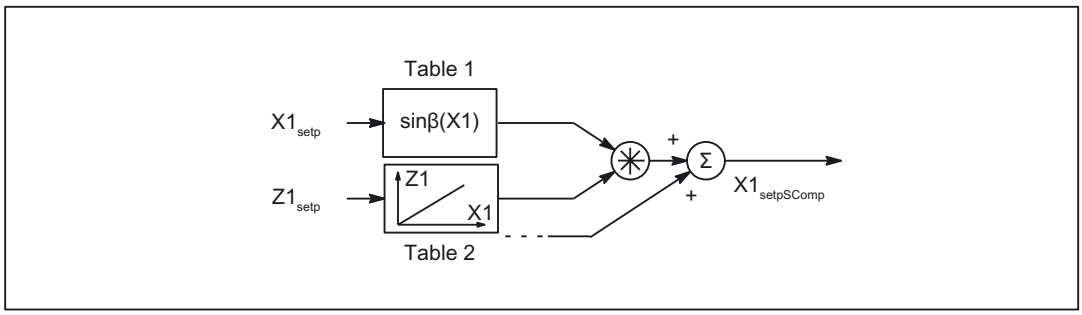


Figure 5-8 Table multiplication

Compensation table 1 (table index = 0) describes the reaction of axis X1 on axis X1 (sine of the position-dependent tilting angle β(X1)).

Compensation table 2 (table index = 1) describes the reaction of axis Z1 on axis X1 (linear).

In table 1, the multiplication of table 1 (index = 0) with table 2 is to be selected:

$AN_CEC_MULT_BY_TABLE[0] = 2

Input of compensation values in a grid structure

The compensation values of the Z axis sag on flat bed machines are often measured in practice at various points as a function of the X and Y coordinates. Under these preconditions, it is useful to enter the measured compensation values according to a grid-type distribution. The interpolation points with the relevant compensation values are positioned at the intersections of the grid (X-Y plane). Compensation values between these interpolation points are interpolated linearly by the control.

The following example explains in more detail how sag and angularity compensation can be implemented by a grid of 4 x 5 (lines x columns) in size. The size of the whole grid is 2000 x 900 mm². The compensation values are each measured in steps of 500 mm along the X axis and 300 mm along the Y axis.

Note

The following interdependencies apply for the maximum dimension of the grid (number of lines and columns):

- The number of lines depends on the number of axes in the system (dependent on the NCU type).
- The number of columns is dependent on the maximum number of values which can be entered in a compensation table (up to a total of 2000 values).

⚠ CAUTION

The number of lines and columns is set via the following machine data:

MD18342 $_MN_MM_CEC_MAX_POINTS (maximum number of interpolation points for sag compensation)

This machine data is memory-configuring!

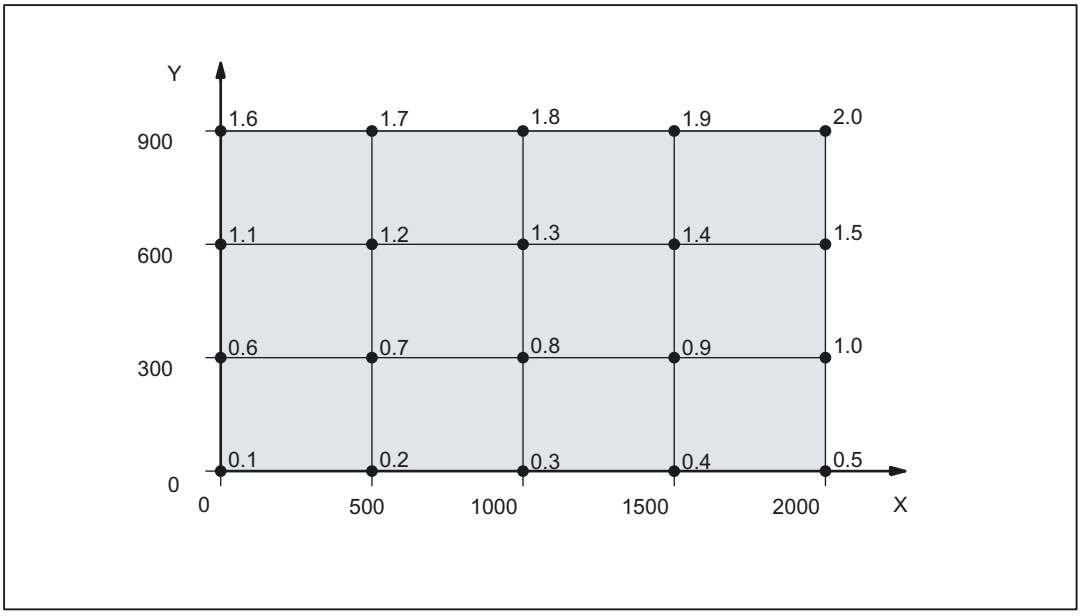


Figure 5-9 Compensation values of Z axis with chessboard-like distribution of X-Y plane

The application example can be realized with the following part program code:

```
$MA_CEC_ENABLE[Z1]           = FALSE      ; Deactivate compensation
                                          ; by setting to FALSE.
                                          ; The table values can then be
                                          ; altered without generation of
                                          ; alarm 17070.
NEWCONF                                   ; Activate $MA_CEC_ENABLE
;Define values f_i(x) in the f tables:
;Function values f_1(x) for table with index [0]
$AN_CEC [0,0]                =0.1
$AN_CEC [0,1]                =0.2
$AN_CEC [0,2]                =0.3
$AN_CEC [0,3]                =0.4
$AN_CEC [0,4]                =0.5

;Function values f_2(x) for table with index [1]
$AN_CEC [1,0]                =0.6
$AN_CEC [1,1]                =0.7
$AN_CEC [1,2]                =0.8
$AN_CEC [1,3]                =0.9
$AN_CEC [1,4]                =1.0

;Function values f_3(x) for table with index [2]
$AN_CEC [2,0]                =1.1
$AN_CEC [2,1]                =1.2
$AN_CEC [2,2]                =1.3
$AN_CEC [2,3]                =1.4
$AN_CEC [2,4]                =1.5

;Function values f_4(x) for table with index [3]
$AN_CEC [3,0]                =1.6
$AN_CEC [3,1]                =1.7
$AN_CEC [3,2]                =1.8
$AN_CEC [3,3]                =1.9
$AN_CEC [3,4]                =2.0

;Enable evaluation of f tables with compensation values
$SN_CEC_TABLE_ENABLE[0]      =TRUE
$SN_CEC_TABLE_ENABLE[1]      =TRUE
$SN_CEC_TABLE_ENABLE[2]      =TRUE
$SN_CEC_TABLE_ENABLE[3]      =TRUE

;Define weighting factor of f tables
$SN_CEC_TABLE_WEIGHT[0]      =1.0
```

```
$SN_CEC_TABLE_WEIGHT[1]      =1.0
$SN_CEC_TABLE_WEIGHT[2]      =1.0
$SN_CEC_TABLE_WEIGHT[3]      =1.0

;Changes to the following table parameters do not take effect until
;a Power On
;Define base axis X1
$AN_CEC_INPUT_AXIS[0]        =(X1)
$AN_CEC_INPUT_AXIS[1]        =(X1)
$AN_CEC_INPUT_AXIS[2]        =(X1)
$AN_CEC_INPUT_AXIS[3]        =(X1)

;Define compensation axis Z1
$AN_CEC_OUTPUT_AXIS[0]       =(Z1)
$AN_CEC_OUTPUT_AXIS[1]       =(Z1)
$AN_CEC_OUTPUT_AXIS[2]       =(Z1)
$AN_CEC_OUTPUT_AXIS[3]       =(Z1)

;Define distance between interpolation points for compensation values in f tables
$AN_CEC_STEP[0]              =500.0
$AN_CEC_STEP[1]              =500.0
$AN_CEC_STEP[2]              =500.0
$AN_CEC_STEP[3]              =500.0

;Compensation starts at X1=0
$AN_CEC_MIN[0]               =0.0
$AN_CEC_MIN[1]               =0.0
$AN_CEC_MIN[2]               =0.0
$AN_CEC_MIN[3]               =0.0

;Compensation ends at X1=2000
$AN_CEC_MAX[0]               =2000.0
$AN_CEC_MAX[1]               =2000.0
$AN_CEC_MAX[2]               =2000.0
$AN_CEC_MAX[3]               =2000.0
;Values of f tables with index [t1] are multiplied by values in g tables
;by the number [t2]
;in accordance with the rule of calculation specified above
$AN_CEC_MULT_BY_TABLE[0]        =5
$AN_CEC_MULT_BY_TABLE[1]        =6
$AN_CEC_MULT_BY_TABLE[2]        =7
$AN_CEC_MULT_BY_TABLE[3]        =8

;Define the g table values for g_i(y) :
;Function values g_1(x) for table with index [4]
```

```
$AN_CEC [4,0]                    =1.0
$AN_CEC [4,1]                    =0.0
$AN_CEC [4,2]                    =0.0
$AN_CEC [4,3]                    =0.0

;Function values g_2(x) for table with index [5]
$AN_CEC [5,0]                    =0.0
$AN_CEC [5,1]                    =1.0
$AN_CEC [5,2]                    =0.0
$AN_CEC [5,3]                    =0.0

;Function values g_3(x) for table with index [6]
$AN_CEC [6,0]                    =0.0
$AN_CEC [6,1]                    =0.0
$AN_CEC [6,2]                    =1.0
$AN_CEC [6,3]                    =0.0

;Function values g_4(x) for table with index [7]
$AN_CEC [7,0]                    =0.0
$AN_CEC [7,1]                    =0.0
$AN_CEC [7,2]                    =0.0
$AN_CEC [7,3]                    =1.0

;Enable evaluation of g tables with compensation values
$SN_CEC_TABLE_ENABLE[4]          =TRUE
$SN_CEC_TABLE_ENABLE[5]          =TRUE
$SN_CEC_TABLE_ENABLE[6]          =TRUE
$SN_CEC_TABLE_ENABLE[7]          =TRUE

;Define weighting factor for g tables
$SN_CEC_TABLE_WEIGHT[4]          =1.0
$SN_CEC_TABLE_WEIGHT[5]          =1.0
$SN_CEC_TABLE_WEIGHT[6]          =1.0
$SN_CEC_TABLE_WEIGHT[7]          =1.0

;Changes to the following table parameters do not take effect until
;a Power On
;Define basic axis Y1
$AN_CEC_INPUT_AXIS[4]                    =(Y1)
$AN_CEC_INPUT_AXIS[5]                    =(Y1)
$AN_CEC_INPUT_AXIS[6]                    =(Y1)
$AN_CEC_INPUT_AXIS[7]                    =(Y1)

;Define compensation axis Z1
$AN_CEC_OUTPUT_AXIS[4]                   =(Z1)
```

```
$AN_CEC_OUTPUT_AXIS[5]                  =(Z1)
$AN_CEC_OUTPUT_AXIS[6]                  =(Z1)
$AN_CEC_OUTPUT_AXIS[7]                  =(Z1)
;Define distance between interpolation points for compensation values in g tables
$AN_CEC_STEP[4]                         =300.0
$AN_CEC_STEP[5]                         =300.0
$AN_CEC_STEP[6]                         =300.0
$AN_CEC_STEP[7]                         =300.0

;Compensation starts at Y1=0
$AN_CEC_MIN[4]                          =0.0
$AN_CEC_MIN[5]                          =0.0
$AN_CEC_MIN[6]                          =0.0
$AN_CEC_MIN[7]                          =0.0

;Compensation ends at Y1=900
$AN_CEC_MAX[4]                          =900.0
$AN_CEC_MAX[5]                          =900.0
$AN_CEC_MAX[6]                          =900.0
$AN_CEC_MAX[7]                          =900.0
$MA_CEC_ENABLE[Z1]           =TRUE      ;Activate compensation again
NEWCONF

;Carry out a program test to check whether the
;compensation is effective
G01 F1000 X0 X0 Z0 G90
R1=0 R2=0
LOOP_Y:
LOOP_X:
STOPRE
X=R1 Y=R2
M0                          ; Wait to check the CEC value
R1=R1+500
IF R1 <=2000 GOTOB LOOP_X
R1=0
R2=R2+300
IF R2<=900 GOTOB LOOP_Y
```

Note

You can read the compensation value under variable "Sag + temperature compensation" on the operator interface. To do so, select softkey "Diagnosis" followed by softkey "Service axis". The currently effective compensation value is displayed next to the "Sag + temperature compensation" variable.

```
;to prepare the table configuration, the Power On
;machine data are set
;cec.md:
;Set option data for commissioning
;Define the number of interpolation points in the compensation tables
;Machine data is memory-configuring
$MN_MM_CEC_MAX_POINTS[0]=5
$MN_MM_CEC_MAX_POINTS[1]=5
$MN_MM_CEC_MAX_POINTS[2]=5
$MN_MM_CEC_MAX_POINTS[3]=5
$MN_MM_CEC_MAX_POINTS[4]=4
$MN_MM_CEC_MAX_POINTS[5]=4
$MN_MM_CEC_MAX_POINTS[6]=4
$MN_MM_CEC_MAX_POINTS[7]=4
$MA_CEC_MAX_SUM[AX3]=10.0          ; Define the maximum
                                   ; total compensation value
$MA_CEC_MAX_VELO[AX3]=100.0        ; Limit the maximum changes in the
                                   ; total compensation value
M17
```

Explanation:

The compensation values cannot be entered directly as a 2-dimensional grid. Compensation tables in which the compensation values are entered must be created first.

A compensation table contains the compensation values of one line (four lines in the example, i.e. four compensation tables). The compensation values 0.1 to 0.5 are entered in the first table in the example, the compensation values 0.6 to 1.0 in the second table, and so on. The compensation tables are referred to below as f tables and their values as f_i(x) (i=number of table).

The compensation values of f tables are evaluated by multiplying them by other tables. The latter are referred to below as g tables and their values as g_i(y). The number of f tables and g tables is equal (four in the example).

In g tables, one compensation value in each table is set to 1 and all the others to 0. The position of compensation value 1 within the table is determined by the table number. In the first g table, compensation value 1 is positioned at the first interpolation point and, in the second g table, at the second interpolation point, etc. By multiplying g tables by f tables, the correct compensation value in each f table is selected by multiplying it by 1. All irrelevant compensation values are concealed through multiplication by 0.

Using this scheme, compensation value D_z at position (x/y) is calculated according to the following equation:

$D_z(x/y)=f_1(x)*g_1(y) + f_2(x)*g_2(y) + ...$

When the compensation value for the actual position of the machine spindle is calculated, the f table values are multiplied by the g table values according to this rule.

Applied to the example, this means, for instance, that compensation value $D_z(500/300)$ is calculated by multiplying each of the function values f_i(500) in the f tables by the function values g_i(300) in the g tables:

$D_z(500/300) = f_1(1000)*g_1(300) + f_2(1000)*g_2(300) + f_3(1000)*g_3(300) + f_4(1000)*g_4(300)$

$D_z(500/300) = 0.2*0 + 0.7*1 + 1.2*0 + 1.7*0 = 0.7$

5.4.4 Direction-dependent leadscrew error compensation

5.4.4.1 Description of functions

If the direction-dependent differences at the compensation points are excessively high, for an inconsistent backlash or for extremely high demands placed on the precision, then it may be necessary to apply direction-dependent compensation of the leadscrew error or measuring system error (for direct position sensing).

"Direction-dependent leadscrew error compensation" ("direction-dependent LEC" or also "bidirectional LEC")

For the "direction-dependent LEC", two compensation tables can be used for each axis: One table for the plus traversing direction and one for the minus traversing direction. The error deviation at the particular compensation point is entered directly into the table (error = compensation value). The control then automatically compensates with the opposite sign. This always automatically results in a linear interpolation between adjacent interpolation points so that no step-like position changes can occur as a result of the compensation.

Preconditions/activation

The "direction-dependent LEC" is implemented in the SINUMERIK control as a special case of "sag compensation". This is the reason that the preconditions and conditions of "sag compensation" apply (see "Compensation of sag and angularity errors (Page 364)").

The activation of the compensation can be checked using a reference measurement, e.g. using the laser interferometer or in the simplest case, using the service display of the particular axis.

NOTICE
If the "direction-dependent LEC" is used in parallel to the sag compensation and compensation of the angularity, then the secondary conditions of these functions must be taken into consideration together, e.g. the assignment of tables <t> to the particular function.

5.4.4.2 Commissioning

Measuring the error or compensation values

Based on the procedure for the direction independent LEC (see"Compensation of leadscrew errors and measuring system errors (Page 358)") for the direction-dependent LEC, the **direction-dependent error curve for each axis** is determined using a suitable measuring device (e.g. laser interferometer). To do this, an appropriate NC program with measurement points and wait times must be generated (see the program "BI_SSFK_MESS_AX1_X.MPF" in the "Example (Page 386)"). The software of the various measuring devices offers, in this case, different support for the practical implementation with the SINUMERIK control. This is the reason that this operation is only described here generally and for specific controls.

The leadscrew errors (= setpoint position - actual position) of the freely selected interpolation points (measuring points) are assigned from the direction-dependent error curves.

The positive direction is obtained with increasing position values.

The measurement should only be performed when commissioning for the first time if the traversing direction of the axes has already been adapted to the coordinate system of the machine.

Commissioning the direction-dependent LEC

1. Enable the "sag compensation, multidimensional" option.

   Note

   Take into account the licensing procedure!

2. Define the number of compensation interpolation points (see also "Compensation of sag and angularity errors: Commissioning (Page 368)")

   Two compensation tables for the positive **and** negative traversing directions should be assigned to each axis and the number of compensation interpolation points defined using the following machine data:

   MD18342 $MN_MM_CEC_MAX_POINTS[<t>] (maximum number of interpolation points for sag compensation)

   with: <t> = Index of compensation table

   Permissible range: 0 ≤ t < 2 * maximum number of axes

   Example:

   MD18342 [0] = 11; 11 interpolation points for the 1st table, e.g. positive traversing direction, X axis

   MD18342 [1] = 11; 11 interpolation points for the 2nd table, e.g. negative traversing direction, X axis

   MD18342 [2] = 21; 21 interpolation points for the 3rd table, e.g. positive traversing direction, Y axis

   MD18342 [3] = 21; 21 interpolation points for the 4th table, e.g. positive traversing direction, Y axis

   …

   MD18342 [61] = ...; number of interpolation points for the 62nd table

   NOTICE

   ALARM 4400 is output when changing MD18342:

   "Reorganization of the buffered memory!"

   In order that an automatic memory configuration is possible but keeping all of the data that has been entered up until now, **no** system boot (POWER ON) must be executed without first performing a series machine startup.

3. Perform the series machine startup:
   - Generate an NC archive with the entries in MD18342 [<t>].
   - Read-in the archive again.

     → The NC memory is automatically configured.

   The compensation values can now be written to the memory locations for the tables.

4. Generate the tables with compensation values for the particular axes and traversing directions as NC program (refer to the section "Measuring the error or compensation values").

5. Execute the program with compensation values in the control.

   AUTOMATIC mode > select program > NC start

   Note

   Each time before reading-in the compensation tables, the following parameters should always be set to **0** and then to activate, always be set to **1**:

   MD32710 $MA_CEC_ENABLE[<AXi>] (enable sag compensation) = **0** → **1**

   SD41300 $SN_CEC_TABLE _ENABLE[<t>] (enable the compensation table) = **0** → **1**

   The backlash should always be set to **0**:

   MD32450 $MA_BACKLASH [<e>] (backlash) = **0**

   with: <e> = Position measuring system

   These automations are automated by using the program template "BI_SSFK_TAB_AX1_X.MPF" (see "Example (Page 386)"). When manually entering machine data, the generally applicable "Activate MD" or "Reset" should be observed.

6. POWER ON (warm restart).
7. Now, comparative measurements can be made using the laser interferometer.
8. To further improve the compensation results, it is also conceivable to correct individual compensation values in the program. A POWER ON is no longer necessary when reading-in the table again.

Note

Sequence for SINUMERIK 828D

For SINUMERIK 828D, steps **2** and **3** are eliminated. This is because when the "sag compensation, multi-dimensional" option is enabled, 8 tables each with 128 interpolation points per table for the compensation immediately become available. This cannot be extended!

Note

As described in step **5**, the compensation table is downloaded into the program memory as an executable program and is then transferred into the previously configured memory area of the control using an NC start. This procedure can be repeated for each table to ensure transparency. However, it is also possible to download all tables in an initialization step. The compensation values become active after MD32710[<AXi>] = 1 and a mandatory power POWER ON.

Note

NC_CEC.INI

The "NC_CEC.INI" file copied via "Commissioning" > "System data" (from the folder "NC active data" > "sag angularity comp") includes all negotiated sag/angularity and direction-dependent LEC tables.

Table parameters

The position-related compensations for the particular direction as well as additional table parameters in the form of system variables should be saved in the compensation table:

- $AN_CEC[<t>,<N>] (compensation value for interpolation point <N> of compensation table [<t>])
- $AN_CEC_INPUT_AXIS[<t>] (basic axis)
- $AN_CEC_OUTPUT_AXIS[<t>] (compensation axis)

  Note

  For the "direction-dependent LEC", the basis and compensation axis are **identical**.

- $AN_CEC_STEP[<t>] (interpolation point distance)
- $AN_CEC_MIN[<t>] (initial position)
- $AN_CEC_MAX[<t>] (end position)
- $AN_CEC_DIRECTION[<t>] (direction-dependent compensation)

  This system variable is used to set whether the compensation table [<t>] should apply to both positive and negative traversing directions of the basic axis:

  – $AN_CEC_DIRECTION[<t>] = 1:

    Table applies only to the positive traversing direction of the base axis

  – $AN_CEC_DIRECTION[<t>] = -1:

    Table applies only to the negative traversing direction of the base axis

  Note

  The setting $AN_CEC_DIRECTION[<t>] = 0 (table is effective for both traversing directions of the basic axis) is **not** relevant for the "direction-dependent LEC".

- $AN_CEC_IS_MODULO[<t>] (compensation with modulo function)

Note

For a detailed description of these system variables, see "Compensation of sag and angularity error: Commissioning (Page 368)".

System of units

See "Compensation of sag and angularity errors: Commissioning (Page 368)".

Monitoring

See "Compensation of sag and angularity errors: Commissioning (Page 368)".

5.4.4.3 Example

The direction-dependent compensation tables of the X axis are shown in detail for a three-axis machine in the fallowing example:

Configuration

Number of compensation interpolation points:

MD18342 $MN_MM_CEC_MAX_POINTS[**0**] = 11 (Table **1**: Axis **X**, **positive** traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**1**] = 11 (Table **2**: Axis **X**, **negative** traversing direction)

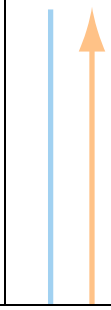
Note

It is not necessary to define the number of interpolation points for SINUMERIK 828D, as, when enabling the "sag compensation, multi-dimensional" option, immediately 8 tables each with 128 interpolation points per table for the compensation are available. This cannot be extended!

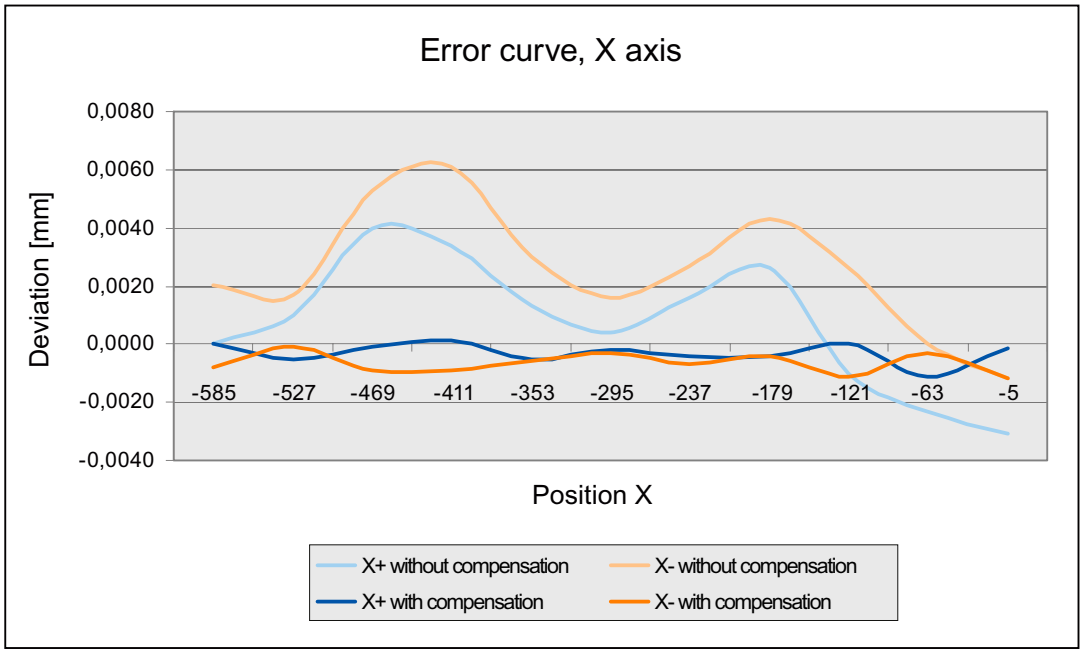
Interpolation points

Table <t>	[0,<N>]										
Number of interpolation points	MD18342 $MN_MM_CEC_MAX_POINTS[0] = **11**										
Interpolation point <N>	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Position X	-585	-527	-469	-411	-353	-295	-237	-179	-121	-63	-5

Measurement

			Setpoint position	Deviation		Checking measurement	
	Position	Comp. No.	Measurement position [mm]	Direction + [mm]	Direction - [mm]	Direction + [mm]	Direction - [mm]
$AC_CEC_MIN[<t>]	-585	0	-585	0.0000	0.0020	0.0000	-0.0008
		1	-527	0.0010	0.0017	-0.0005	-0.0001
		2	-469	0.0040	0.0053	-0.0001	-0.0009
		3	-411	0.0034	0.0061	0.0001	-0.0009
		4	-353	0.0013	0.0030	-0.0005	-0.0006
		5	-295	0.0004	0.0016	-0.0002	-0.0003
		6	-237	0.0016	0.0027	-0.0004	-0.0007
		7	-179	0.0026	0.0043	-0.0004	-0.0004
		8	-121	-0.0010	0.0026	0.0000	-0.0011

		9	-63		-0.0023		0.0000		-0.0011	-0.0003
$AC_CEC_MAX[<t>]	-5	10	-5		-0.0031		-0.0012		-0.0001	-0.0012



Programming

The following program "BI_SSFK_TAB_AX1_X.MPF" includes the value assignments for the parameters of the two compensation tables (positive and negative traversing direction) of the X axis:

```
;direction-dependent LEC
;1st axis MX1
;Table 1 - positive traversing direction
;Table 2 - negative traversing direction
;------------------------------------------------------------------------------
```

```
CHANDATA(1)
$MA_CEC_ENABLE[AX1]=0                ;compensation OFF
$SN_CEC_TABLE_ENABLE[0]=0            ;lock Table 1
$SN_CEC_TABLE_ENABLE[1]=0            ;lock Table 2
NEWCONF
;------------------------------------------------------------------------------
$AN_CEC[0,0]=0                       ;1st compensation value (interpolation point 0)
$AN_CEC[0,1]=0.001                   ;2nd compensation value (interpolation point 1)
$AN_CEC[0,2]=0.004                   ;3rd compensation value (interpolation point 2)
$AN_CEC[0,3]=0.0034                  ;4th compensation value (interpolation point 3)
$AN_CEC[0,4]=0.0013                  ;5th compensation value (interpolation point 4)
$AN_CEC[0,5]=0.0004                  ;6th compensation value (interpolation point 5)
$AN_CEC[0,6]=0.0016                  ;7th compensation value (interpolation point 6)
$AN_CEC[0,7]=0.0026                  ;8th compensation value (interpolation point 7)
$AN_CEC[0,8]=-0.001                  ;9th compensation value (interpolation point 8)
$AN_CEC[0,9]=-0.0023                 ;10th compensation value (interpolation point 9)
$AN_CEC[0,10]=-0.0031                ;last compensation value (interpolation point 10)
$AN_CEC_INPUT_AXIS[0]=(AX1)          ;base axis
$AN_CEC_OUTPUT_AXIS[0]=(AX1)         ;compensation axis
$AN_CEC_STEP[0]=58.0                 ;interpolation point distance
$AN_CEC_MIN[0]=-585.0                ;compensation starts
$AN_CEC_MAX[0]=-5.0                  ;compensation ends
$AN_CEC_DIRECTION[0]=1               ;Table applies for positive traversing directions
$AN_CEC_MULT_BY_TABLE[0]=0           ;no multiplication (not relevant here)
$AN_CEC_IS_MODULO[0]=0               ;compensation without modulo function
;------------------------------------------------------------------------------
$AN_CEC[1,0]=0.002                   ;(interpolation point 0)
$AN_CEC[1,1]=0.0017                  ;(interpolation point 1)
$AN_CEC[1,2]=0.0053                  ;(interpolation point 2)
$AN_CEC[1,3]=0.0061                  ;(interpolation point 3)
$AN_CEC[1,4]=0.003                   ;(interpolation point 4)
$AN_CEC[1,5]=0.0016                  ;(interpolation point 5)
$AN_CEC[1,6]=0.0027                  ;(interpolation point 6)
$AN_CEC[1,7]=0.0043                  ;(interpolation point 7)
$AN_CEC[1,8]=0.0026                  ;(interpolation point 8)
$AN_CEC[1,9]=0.000                   ;(interpolation point 9)
$AN_CEC[1,10]=-0.0012                ;(interpolation point 10)
$AN_CEC_INPUT_AXIS[1]=(AX1)          ;base axis
$AN_CEC_OUTPUT_AXIS[1]=(AX1)         ;compensation axis
$AN_CEC_STEP[1]=58.                  ;interpolation point distance
$AN_CEC_MIN[1]=-585.0                ;compensation starts
$AN_CEC_MAX[1]=-5.0                  ;compensation ends
$AN_CEC_DIRECTION[1]=-1              ;Table applies for negative traversing directions
$AN_CEC_MULT_BY_TABLE[1]=0           ;no multiplication (not relevant here)
$AN_CEC_IS_MODULO[1]=0               ;compensation without modulo function (only for rotary axes)
```

```
;-------------------------------------------------------------------------------
$MA_CEC_ENABLE[AX1]=1                ;compensation ON
$SN_CEC_TABLE_ENABLE[0]=1            ;enable Table 1
$SN_CEC_TABLE_ENABLE[1]=1            ;enable Table 2
NEWCONF
M17
```

Additional tables can be set-up, e.g. for axes Y and Z:

MD18342 $MN_MM_CEC_MAX_POINTS[**2**] = 90 (Table **3**: Axis **Y**, positive traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**3**] = 90 (Table **4**: Axis **Y**, negative traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**4**] = 50 (Table **5**: Axis **Z**, positive traversing direction)

MD18342 $MN_MM_CEC_MAX_POINTS[**5**] = 50 (Table **6**: Axis **Z**, negative traversing direction)

5.4.5 Extension of the sag compensation with NCU link - only 840D sl

Application

If a system is operated with NCU link, any number of axes of the NCU link grouping can be compensated mutually. The two axes that are coupled via sag compensation must be interpolated on one NCU.

See also

- Several Operator Panels on Several NCUs, Distributed Systems (B3); Chapter: NCU-Link
- Several Operator Panels on Several NCUs, Distributed Systems (B3); Chapter: Axis Container

Function

The parameterization of the sag compensation function is done by setting system variables of the form: $AN_CEC ...

These system variables are normally set via a part program that processes the NCK in a certain channel. The channel axis identifier can be used in the variables $AN_CEC_OUTPUT_AXIS or $AN_CEC_INPUT_AXIS. This way, each axis of the channel can be addressed, even if it is in a different NCU.

A program in Channel 2 can couple Axis ZZ with Axis XX via the following setting (the setting is valid for the table with the number 0):

$AN_CEC_INPUT_AXIS[0] = (XX)

$AN_CEC_OUTPUT_AXIS[0] = (ZZ)

This way AX3 on NCU-1 is "coupled" with AX2 on NCU-2 (see configuration 1).

The following variants can be used to parameterize if the axes to be coupled are on different channels:

- **Version 1: "Programming with channel axis identifier"**:

  Two different part programs TP1 and TP2 are created, they are then processed in different channels.

  Axis "ZZ" is coupled to "XR":

  View from the part program TP1 in Channel 1:

  ```
  $AN_CEC_INPUT_AXIS[0] = (XR)
  ```

  View from the part program TP2 in Channel 2:

  ```
  $AN_CEC_OUTPUT_AXIS[0] = (ZZ)
  ```

  Axis AX2 on NCU2 is coupled with Axis AX1 on NCU1 upon restart after TP1 is executed in Channel 1 and TP2 is executed in Channel 2.

- **Version 2: "Programming with machine axis identifier"**:

  One part program is created that runs in any convenient channel of NCU1 and specifies the machine axis names together with the NCU numbers.

  Axis "ZZ" is coupled to "XR":

  ```
  $AN_CEC_INPUT_NCU[0]=1
  $AN_CEC_INPUT_AXIS[0] = (AX1)
  $AN_CEC_OUTPUT_NCU[0]=2
  $AN_CEC_OUTPUT_AXIS[0] = (AX2)
  ```

  The NCK monitors whether the axes on the local NCU have actually been interpolated, i.e., there is a channel that can program these axes. The local NCU is always the NCU on which the part program runs.

  The following axes are allowed for NCU1 as input or output axes in Configuration 1: NC1_AX1, NC1_AX3, NC1_AX4, NC1_AX5, NC2_AX2, NC2_AX6

  The data backup from the NCK always delivers the compensation data from the "machine axis indentifier" view.

NOTICE
The NCU no. is to be programmed before the axis identifier. A sag compensation between NC1_AX1 and NC1_AX2 is not possible.

Assignment of the axes

The assignment of the input and output axes is done via the following system variables:

$AN_CEC_INPUT_NCU	and	$AN_CEC_INPUT_AXIS
$AN_CEC_OUTPUT_NCU	and	$AN_CEC_OUTPUT_AXIS

The system variables become effective only after a restart.

Data backup is always undertaken with machine axis identifiers.

Note

The sag compensation can couple the axis only on one NCU, which can also be traversed from this NCU either via the part program or via a synchronized action.

These variables are set optionally if the axes (input and output) are not available on the local NCU. If one uses a channel axis identifier while programming $AN_CEC_INPUT_AXIS and $AN_CEC_OUTPUT_AXIS, then the system variables $AN_CEC_INPUT_NCU and $AN_CEC_OUTPUT_NCU become irrelevant.

The control checks whether the two axes can be interpolated from this NCU, i.e., a program can traverse the axes on this NCU. The axes can be assigned to different channels. Two axes belonging to different NCUs can also be compensated. Otherwise the control rejects it with Alarm 17040.

Both axes of compensation must be interpolated on one NCU, i.e., there may be one or two part programs that traverse the input and output axes on an NCU.

Axis container

The axis container is a grouping of similar axes. An axis from the group can be assigned to a channel axis. The assignment is variable, so that the axis in the channel always gets a new axis from the group assigned to it in the course of time. Thus, the part program can be programmed with one axis and it can gradually move different axes.

Example:

Four spindles are arranged on a drum. Each spindle carries a tool of the turning machine and it rotates the drum by 90 degrees in each cycle. The tools are transported from one machining station to the next in this way. The channel of the machining station must program only one spindle, though a new spindle is always changed. This is an axis container rotation.

The sag compensation can be combined with the axis container if it is in the basic position, i.e., $AN_AXCTAS == 0. Otherwise the programming is rejected with Alarm 17040.

"YY" is to be coupled to "XX" (see Configuration 2):

1. Programming with channel axis identifier

```
$AN_CEC_INPUT_AXIS[0] = (XX)
$AN_CEC_OUTPUT_AXIS[0] = (YY)
```

2. Programming with machine axis identifier

```
$AN_CEC_INPUT_NCU[0]=1          ; optional ...
$AN_CEC_INPUT_AXIS[0] = (AX2)
$AN_CEC_OUTPUT_NCU[0]=2
$AN_CEC_OUTPUT_AXIS[0] = (AX2
```

This couples Axis AX2 of NCU1 with Axis 2 of NCU2.

NOTICE
YY is coupled to XX with each container rotation, there is a different axis behind YY now: YY "AX5 of NCU-1" is in configuration 3. Other real axes are coupled after the rotation in this way: In this example, AX-5 of NCU-1 is coupled to AX-2 of NCU-1.

As a rule:

The coupling is created between two axes from the LAI layer so that other axes participate in the coupling after each axis container rotation. A new table must be activated for each container rotation to undertake a coupling exactly between two real axes.

Configuration example

The following figures (Configuration 1, Configuration 2 and Configuration 3) illustrate the axis configurations of an NCU link that is assembled from two NCUs.

The two channels CHAN-1 and CHAN-2 of NCU-1 are displayed in Configuration 1. Here, the channel axis names that are defined via the machine data $MC_AXCONF_CHANAX_NAME_TAB are entered. The channel configuration of the second NCU is not displayed.

All the axes interpolated by this NCU are compiled in the "Logical NCK machine axis image" (LAI layer). The assignment between channel and MCS axis layer is done via $MC_AXCONF_MACHAX_USED.

The assignment between the "Logical NCK machine axis image" and the real axes is undertaken via the machine data $MN_AXCONF_LOGICMACHAX_TAB. If ones pursues the connecting line that starts with channel axis ZZ, one ends at Axis AX2 on NCU-2, i.e., to traverse the 2nd axis of NCU 2, the following instruction must be programmed in the 2nd channel of NCU 1: `"N2040 POS[ZZ]=10 FA[ZZ]=1000"`

Configuration 2 and Configuration 3 extend the figure of Configuration 1 by one axis container (CT1) that is set with machine data $MN_AXCT_AXCONF_ASSIGN_TAB1. The axis container is an overlapping object, i.e., each axis container exists only once for the whole NCU cluster.

For NCU 1, the participants in the axis container are channel axes YR and YY; the two channel axes from NCU2 are not displayed. The container contains the real axes NC1_AX5, NC1_AX6, NC2_AX1 and NC2_AX2. Container YR connects with NC2_AX1 and YY connects with NC2_AX2 during the ramp up. In Configuration 3, the container has rotated, i.e., the connection structure has changed. YR is now connected to NC2_AX2 and YY is connected to NC1_AX5.

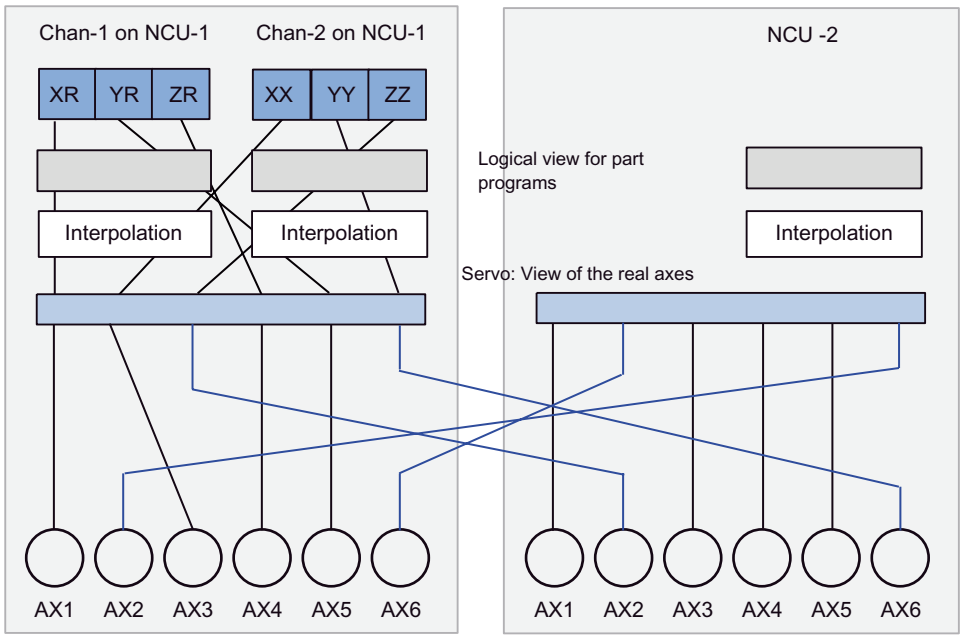


Figure 5-10 Configuration 1: NCU link from channel to real axis

Machine data of Configuration 1

```
; ########## NCU1 ##########
$MN_NCU_LINKNO = 1
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3

$MN_ASSIGN_CHAN_TO_MODE_GROUP[1]=1

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC1_AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "NC1_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC1_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "NC1_AX5"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "NC2_AX6"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=5
$MC_AXCONF_MACHAX_USED[2]=4
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XR"
```

```
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YR"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZR"

CHANDATA(2)
$MC_REFP_NC_START_LOCK=0
$MC_AXCONF_MACHAX_USED[0]=2
$MC_AXCONF_MACHAX_USED[1]=6
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YY"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZZ"
M30

; ########## NCU-2 ##########
$MN_NCU_LINKNO = 2
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3
;
;
;
$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC2_AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "NC1_AX6"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC2_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "NC2_AX5"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "NC1_AX2"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=4
$MC_AXCONF_MACHAX_USED[4]=5
$MC_AXCONF_MACHAX_USED[5]=6
$MC_AXCONF_MACHAX_USED[6]=0
M30
```

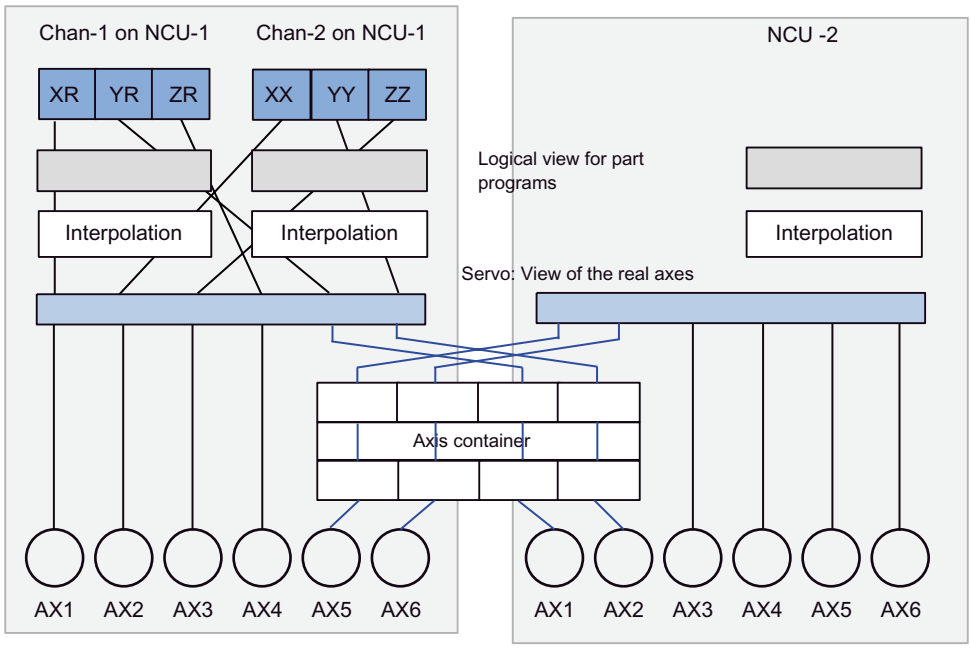


Figure 5-11 Configuration 2: NCU link with axis container in output state

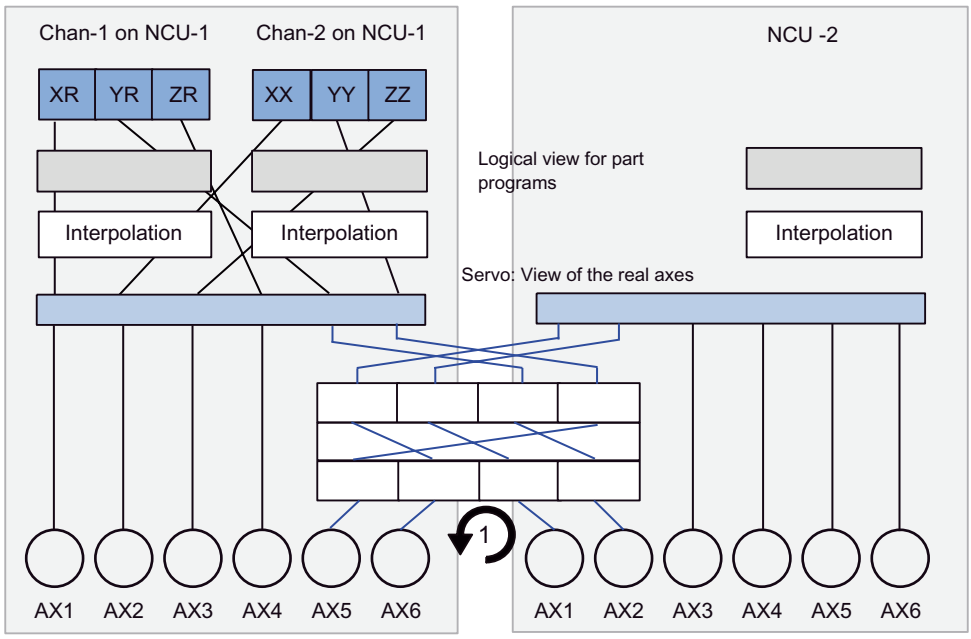


Figure 5-12 Configuration 3: NCU link with axis container in rotary state

Machine data of Configuration 2

```
; ########## NCU1 ##########
$MN_NCU_LINKNO = 1
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3

$MN_ASSIGN_CHAN_TO_MODE_GROUP[1]=1

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "NC1_AX1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "NC1_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX2"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC1_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "CT1_SL3"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "CT1_SL4"

$MN_AXCT_AXCONF_ASSIGN_TAB1[0] = "NC1_AX5"
$MN_AXCT_AXCONF_ASSIGN_TAB1[1] = "NC1_AX6"
$MN_AXCT_AXCONF_ASSIGN_TAB1[2] = "NC2_AX1"
$MN_AXCT_AXCONF_ASSIGN_TAB1[3] = "NC2_AX2"

$SN_AXCT_SWWIDTH[0] = 1

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=5
$MC_AXCONF_MACHAX_USED[2]=4
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XR"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YR"
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZR"

CHANDATA(2)
$MC_REFP_NC_START_LOCK=0
$MC_AXCONF_MACHAX_USED[0]=2
$MC_AXCONF_MACHAX_USED[1]=6
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=0
$MC_AXCONF_MACHAX_USED[4]=0
$MC_AXCONF_MACHAX_USED[5]=0
$MC_AXCONF_CHANAX_NAME_TAB[0] = "XX"
$MC_AXCONF_CHANAX_NAME_TAB[1] = "YY"
```

```
$MC_AXCONF_CHANAX_NAME_TAB[2] = "ZZ"
M30

; ########## NCU-2 ##########
$MN_NCU_LINKNO = 2
$MN_MM_NCU_LINK_MASK = 1
$MN_MM_LINK_NUM_OF_MODULES = 2
$MN_MM_SERVO_FIFO_SIZE = 3

$MN_AXCONF_LOGIC_MACHAX_TAB[0] = "CT1_SL1"
$MN_AXCONF_LOGIC_MACHAX_TAB[1] = "CT1_SL2"
$MN_AXCONF_LOGIC_MACHAX_TAB[2] = "NC2_AX3"
$MN_AXCONF_LOGIC_MACHAX_TAB[3] = "NC2_AX4"
$MN_AXCONF_LOGIC_MACHAX_TAB[4] = "NC2_AX5"
$MN_AXCONF_LOGIC_MACHAX_TAB[5] = "NC2_AX6"

CHANDATA(1)
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
$MC_AXCONF_MACHAX_USED[3]=4
$MC_AXCONF_MACHAX_USED[4]=5
$MC_AXCONF_MACHAX_USED[5]=6
$MC_AXCONF_MACHAX_USED[6]=0
M30
```

5.4.6 Special features of interpolatory compensation

Measuring

The "Measurement" function supplies the compensated actual positions (ideal machine) required by the machine operator or programmer.

TEACH IN

The "TEACH IN" function also uses compensated position values to determine the actual positions to be stored.

Software limit switch

The ideal position values (i.e. the position actual values corrected by the MSEC and backlash compensation functions) are also monitored by the software limit switches.

Position display

The position actual-value display in the machine coordinate system shows the ideal (programmed) actual position value of the axis (ideal machine).

The position actual value determined by the measuring system plus the sum of MSEC and backlash compensation (= position actual value, measuring system 1/2) is displayed the "axis/spindle" service display (operating area "Diagnosis")

Compensation value display

The following compensation values are also output in the "Axis/spindle" service display (operating area "Diagnosis"):

- Absolute compensation value measuring system 1 or 2

  Displayed value corresponds to the total compensation value calculated from MSEC and backlash compensation associated with the actual position of the axis (measuring system 1 or 2).

- Compensation, sag + temperature

  Display value is the sum of the compensation values from sag compensation and temperature compensation for the actual position of the axis.

Reference:
Function Manual, Basic Functions; Diagnostic Tools (D1)

Reference point loss

If the reference point of the basic axis is lost (DB31, ... DBX60.4 or 60.5 = 0), then MSEC and sag compensation functions are deactivated in the axes involved. These are automatically reactivated when the reference point is reached.

Access protection

Currently there is no protection against access to the compensation tables.

Setting servo enables

As a result of the compensation relationship, a traversing movement by the base axis may also cause the compensation axis to move, making it necessary for controller enable signals to be set for these axes (PLC user program). Otherwise the compensation only has a limited effect.

Traversing signal output

The traversing signals in the compensation axis are output every time the compensation function is switched on/off and every time the number of active compensation tables changes.

Any change in the compensation value caused by the base axis motion does not result in output of traversing signals in the compensation axis.

5.5 Dynamic feedforward control (following error compensation)

5.5.1 General properties

Axial following error

The remaining system deviation of the position controller when traversing a machining axis is known as axial following error. Expressed in another way, the axial following error is the difference between the setpoint position and the actual position of the machine axis.

Effects

Particularly during acceleration in contour curvatures, e.g. circles and corners, this following error leads to undesirable, velocity-dependent contour violations.

Compensation

The axial following error can be reduced almost to zero with the help of the "dynamic feedforward control". The function is therefore also called "following error compensation".

Methods

There are two "dynamic feedforward control" methods:

- Speed feedforward control(velocity dependent)
- Torque feedforward control(acceleration dependent)

Note

The torque feedforward control is an option that must be licensed and is only available for SINUMERIK 840D sl. The function is not available for SINUMERIK 828D.

Activation

The feedforward control method is selected and activated using the machine data:

MD32620 $MA_FFW_MODE (feedforward control mode)

Value	Significance
0	No feedforward control
1	Speed feedforward control with PT1 balancing
2	Torque feedforward control with PT1 balancing
3	Speed feedforward control with Tt balancing (standard setting for 840D sl / 840Di sl)
4	Torque feedforward control with Tt balancing

Note

Upgrading 840D sl and 840Di sl

When upgrading SINUMERIK 840 D sl and 840Di sl, new commissioning settings must be entered.

If the feedforward control version MD32620 = 3 was already used, then when upgrading the software, the commissioning setting of MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant, speed control loop for feedforward control) must be re-performed, as the Ti and To values are automatically taken into account. These values must be corrected manually in MD32810.

Activation/deactivation in part program

The following axis-specific machine data can be used to define that the feedforward control for this axis/spindle can be activated and deactivated by the part program:

MD32630 $MA_FFW_ACTIVATION_MODE (activate feedforward control from program)

Value	Significance
0	The feedforward control cannot be activated and deactivated from the part program. This means that the state specified using MD32620 $MA_FFW_MODE is always effective for the axis/spindle.
1	The feedforward control can be activated and deactivated from the part program. The operation becomes active immediately.
2	The feedforward control can be activated and deactivated from the part program. The operation only becomes active the next time that the axis comes to a standstill.

The feedforward control is activated/deactivated from the part program using the operations:

`FFWON`: Feedforward control ON

`FFWOF`: Feedforward control OFF

The default setting (i.e. M30 even after reset) is entered using the channel-specific machine data:

MD20150 $MC_GCODE_RESET_VALUES (initial setting of G groups)

`FFWON`/`FFWOF` is active for all axes/spindles in the axis mode, where:

MD32630 $MA_FFW_ACTIVATION_MODE = 1 (or 2)

and

MD32620 $MA_FFW_MODE = 1, 2, 3 or 4

The identical MD32630 setting should be used for axes that interpolate with each other.

The feedforward control should only be activated or deactivated while the axis/spindle is stationary in the axis mode, in order to prevent jerky motion. Hence the switchover is delayed automatically up to the next standstill through block search stop.

Please note the following in this context:

A block search stop is not effective for command or PLC axes traversing asynchronously to the subprogram processing. To ensure that FFWON/FFWOF only has an effect on the axis/spindle when it is next stationary in the axis mode, you must explicitly set MD32630 = 2 for each axis/spindle in the axis mode (see also "Forward feed control for command- and PLC axes (Page 406)").

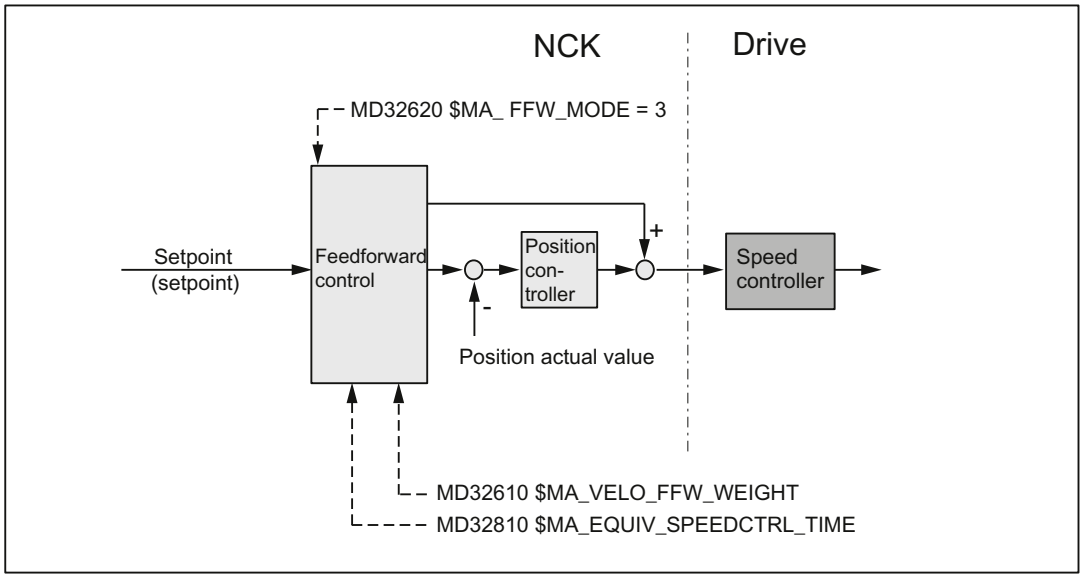
5.5.2 Speed feedforward control

Function

In the case of speed feedforward control, a velocity setpoint is also applied directly to the input of the speed controller. With this value the following error can be reduced to nearly zero (i.e. system deviation is 0) when the velocity is constant.

Commissioning

The following axis-specific parameters must be defined for the speed feedforward control during commissioning:



Equivalent time constant of the speed control loop (MD32810)

The equivalent time constant of the speed control loop must be determined accurately (e.g. graphically from a speed setpoint step response) and entered into the following machine data to correctly set the speed feedforward control:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

Reference:
Function Manual Basic Functions; Velocities, Setpoint/Actual Value Systems, Closed-Loop Control (G2), Chapter: "Optimizing the closed-loop control"

Feedforward control factor for speed feedforward control (MD32610)

The additional velocity setpoint can be weighted using a factor:

MD32610 $MA_VELO_FFW_WEIGHT

Value range: 0 ... 1

"0" means: no feedforward control. As standard, the factor has a value of 1 (≙ 100%).

The factor should remain set at 100%, as this value is the optimum setting for an optimally set control loop for the axis/spindle as well as a precisely determined equivalent time constant of the speed control loop.

Fine adjustment

The speed feedforward control for the particular axis/spindle can be optimized by making slight changes (fine tuning) to the equivalent time constants of the speed control loop (MD32810).

To make this check, the axis/spindle should be traversed at a constant velocity and the response of the machine data changes should be checked in the service display "axis/spindle", "system deviation".

The adjustment criterion for the speed feedforward control is

- When the axis is traveling in a positive direction, the 'system deviation' shows a **positive** value,

  ⇒ The equivalent time constant of the speed control loop is **too small**.

- When the axis is traversing in a positive direction, the 'system deviation' shows a **negative** value.

  ⇒ The equivalent time constant of the speed control loop is **too large**.

A small acceleration and a high feedrate should be chosen so that the values can be easily read on the service display. This produces very long acceleration phases from which it is easy to read off the control deviation.

References:

- Function Manual, Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2), Chapter "Optimization of Control"
- Function Manual, Basic Functions; Diagnostics Tools (D1)

Rate time

A rate time can be defined to output the speed setpoints.

Reference:
SINAMICS S120 Function Manual

Example

Part program to set the equivalent time constants for the X axis

```
Program code                                    Comment
MD32300 $MA_MAX_AX_ACCEL=0,1                    ; Unit: m/s2
MD32000 $MA_MAX_AX_VELO=20000.0                 ; Unit: mm/min
; Part program for setting the equivalent time constant
G1 F20000
FFWON
LOOP:
X1000
X0
GOTOB LOOP
M30
```

5.5.3 Torque feedforward control - only 840D sl (option)

Function

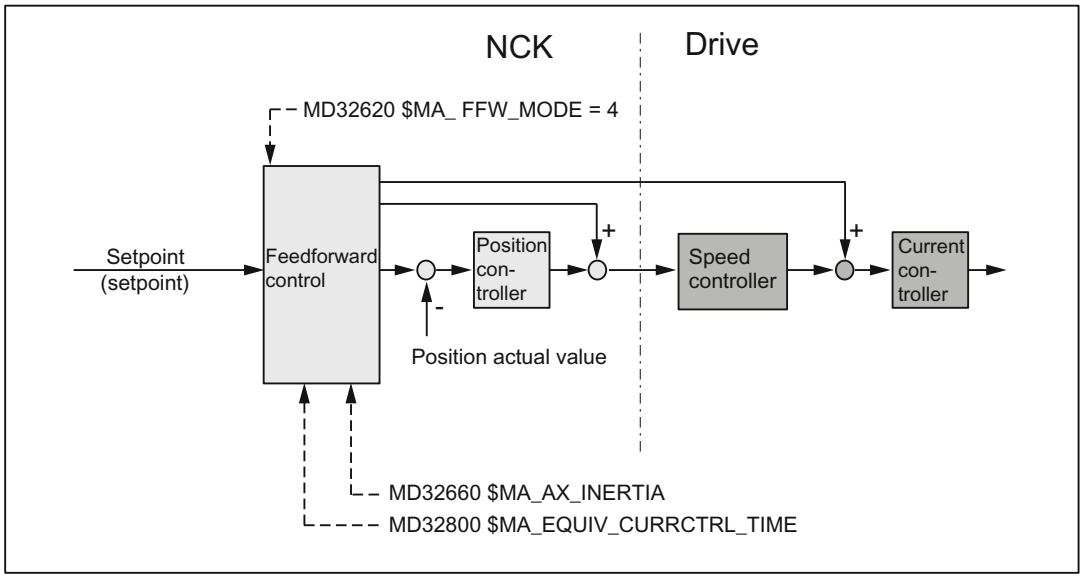
In the case of torque feedforward control, an additional current setpoint proportional to the torque is applied directly to the current controller input. This value is formed using the acceleration and moment of inertia.

Application

Torque feedforward control is required to achieve high contour accuracy where the demands on the dynamics are great. If set correctly, the following error can almost be completely compensated even during high acceleration.

Commissioning

The following axis-specific parameters must be defined during commissioning for torque feedforward control:



Equivalent time constant of the current control loop (MD32800)

The equivalent time constant of the current control loop must be determined accurately (e.g. graphically from step response of the current control loop) and entered in the following machine data in order to correctly set the torque feedforward control:

MD32800 $MA_EQUIV_CURRCTRL_TIME (equivalent time constant current control loop for feedforward control)

Reference:
Function Manual Basic Functions; Velocities, Setpoint/Actual Value Systems, Closed-Loop Control (G2), Chapter: "Optimizing the closed-loop control"

Total moment of inertia of axis (MD32650)

The total moment of inertia (moment of inertia of drive + load referred to the motor shaft) of the axis must be determined and entered in the following machine data:

MD32650 $MA_AX_INERTIA (inertia for torque feedforward control)

Fine adjustment

The torque feedforward control for the particular axis/spindle can be optimized by making slight changes (fine tuning) to the values in MD32800 and MD32650.

To make a check, the following error should be recorded via an analog setpoint output. In addition to traversing at a constant velocity, the following error should be monitored especially when the axis/spindle accelerates.

The adjustment criterion for the torque feedforward control is:

- When the axis is traversing in the positive direction, the recorded following error has a **positive** value.

  ⇒ The value entered for the equivalent time constant of the current control loop or for the moment of inertia of the axis is **too low**.

- When the axis is traversing in the positive direction, the recorded following error has a **negative** value.

  ⇒ The value entered for the equivalent time constant of the current control loop or for the moment of inertia of the axis is **too high**.

Note

As a result of the extremely fast sequences when accelerating, when commissioning the torque feedforward control, the service display cannot be used to check the fine adjustment.

5.5.4 Dynamic response adaptation

Function

For axes that interpolate with one another, but with different axial control loop response times, dynamic response adaptation can be used to achieve identical time responses of all axes to ensure optimum contour accuracy without loss of control quality.

Commissioning

Time constant for dynamic response adaptation (MD32910)

The difference between the equivalent time constants of the "slowest" speed or current control loop and the particular axis should be entered as time constant for the dynamic response adaptation in the following machine data.

MD32910 $MA_DYN_MATCH_TIME (time constant of dynamic response adaptation)

Example:

Equivalent time constants of the speed control loop (MD32810) for active speed feedforward control of axes 1, 2 and 3:

- Axis 1: 2 ms
- Axis 2: 4 ms (dynamically the slowest axis)
- Axis 3: 1 ms

This means that the following values are obtained for the time constant of the dynamic response adaptation MD32910:

- Axis 1: 2 ms
- Axis 2: 0 ms
- Axis 3: 3 ms

Activation (MD32900)

The dynamic response adaptation is only active if the following machine data is set:

MD32900 $MA_DYN_MATCH_ENABLE= 1

Reference

Function Manual, Basic Functions; Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2), Chapter: "Optimizing the closed-loop control"

5.5.5 Forward feed control for command- and PLC axes

Function

For command and PLC axes, it must be prevented that the feedforward control is activated/deactivated at higher velocities as follows:

MD32630 $MA_FFW_ACTIVATION_MODE = 2

With this setting, the `FFWON/FFWOF` operation only becomes active below the stationary velocity (MD36060 $MA_STANDSTILL_VELO_TOL) configured for this particular axis.

If the switchover instruction coincides with an axis motion, the required switchover is executed only in the next stoppage condition of the axis. This avoids the following error being suddenly established/reduced.

NOTICE
A stoppage velocity set to a very high value can lead to the changeover of the feedforward control in the movement. Controls can be activated depending on the existing following error.

Commissioning

We recommend the following procedure when checking the feedforward control for command and PLC axes:

1. Check the stoppage velocity in MD36060.
2. Check the existing following error of the axis in stoppage condition.
3. Setting the changeover condition and activating it:

   MD32630 = 2
4. Traverse axis in the part program using the `POSA` operation.
5. Execute `FFWON` during the axis motion.

6. The K_v factor and following error displayed in the service display "Axis/spindle" must not jump.
7. A higher K_v factor and a lower following error are only obtained for traversing motion following standstill. However, the feedforward control is active only from the stoppage condition.

Essentially the same as when activating the feedforward control, for deactivation, the following applies:

1. Traverse axis in the part program using the `POSA` operation.
2. Execute `FFWOF` during the axis motion.
3. The K_v factor and following error displayed in the service display "Axis/spindle" must not jump.
4. A lower K_v factor and a higher following error are only obtained for traversing motion following standstill. However, the feedforward control is inactive only from the stoppage condition.

Example

In the following program example, axis A is traversed asynchronously to the path. An attempt is made to activate the feedforward control in the channel while traversing. Contrary to the geometry axes X, Y and Z, the feedforward control is not immediately effective for axis A. Here one waits for the stoppage after N60. Axis A then traverses with the feedforward control in N70.

Program code

```
N10 FFWOF
N20 POSA[A]=1000 FA[A]=10000
N30 G4 F1
N40 FFWON
N50 G0 X10 Y10 Z10
N60 WAITP(A)
N70 POSA[A]=1500 FA[A]=10000
N80 WAITP(A)
M30
```

5.5.6 Secondary conditions

Axes that are interpolating axes with one another

Also for axes that interpolate with one another, the feedforward control parameter should be optimally set **for each axis**, i.e. also several axes that are interpolating with one another can have different feedforward control parameters.

Check contour monitoring

As the two equivalent time constants:

- MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)

and

- MD32800 $MA_EQUIV_CURRCTRL_TIME) (equivalent time constant current control loop for feedforward control)

also influence the contour monitoring, this should be subsequently checked.

Reference:
Function Manual, Basic Functions; Axis Monitoring, Protection Zones (A3)

Effect on servo gain factor

When the feedforward control is set correctly, the response to setpoint changes in the controlled system under speed feedforward control is as dynamic as that of the speed control loop or, under torque feedforward control, as that of the current control loop, i.e. the servo gain factor entered into MD32200 $MA_POS_CTRLGAIN hardly has any effect on the control behavior (e.g. corner errors, overshoots, circle/radius errors).

On the other hand, feedforward control does not affect the response to disturbances (synchronism). In this case, the servo gain factor entered in MD32200 is the active factor.

Service display "Servo gain factor"

When a feedforward control is active, the servo gain of the axis (corresponds to servo gain factor active as response to setpoint changes) shown in the service display "axis/spindle" is very high.

5.6 Friction compensation (quadrant error compensation)

5.6.1 General properties

Function

Friction occurs predominantly in the gearing and guideways. Static friction is especially noticeable in the machine axes. Because it takes a greater force to initiate a movement (breakaway) than to continue it, a greater following error occurs at the beginning of a movement.

The same phenomenon occurs on a change of direction where static friction causes a jump in frictional force. If, for example, one axis is accelerated from a negative to a positive velocity, it sticks for a short time as the velocity passes through zero because of the changing friction conditions. With interpolating axes, changing friction conditions can cause contour errors.

Quadrant errors

This behavior is particularly apparent on circular contours on which one axis is moving at maximum path velocity and the other is stationary at quadrant transitions. With the aid of friction compensation these so-called "quadrant errors" can be almost completely eliminated.

General procedure

Measurements on machines have shown that contour errors caused by static friction can be effectively compensated by the injection of an additional setpoint pulse with the correct sign and amplitude.

Methods of friction compensation

One of two friction compensation methods can be selected on the SINUMERIK 840D

(MD32490 $MA_FRICT_COMP_MODE (friction compensation method)):

- Conventional friction compensation

  (MD32490 = 1)

With this type, the intensity of the compensation pulse can be set according to the characteristic as a function of the acceleration. This characteristic must be determined and parameterized during commissioning using the circularity test. The procedure for this is relatively complicated and requires some experience.

- Quadrant error compensation with neural networks

  (option for SINUMERIK 840D)

  (MD32490 = 2)

To simplify commissioning, the compensation characteristic no longer has to be set manually by the commissioning engineer but is determined automatically during a training phase and then stored in the non-volatile user memory.

The neural network can reproduce a compensation curve of much better quality and precision.

The function also allows simple automatic re-optimization directly at the machine.

Circularity test

The friction compensation function (both conventional and neural friction compensation) can be commissioned most easily by means of a circularity test. This is done by following a circular contour, measuring the actual position and representing the deviations from the programmed radius (especially at the quadrant transition points) graphically. The measurements are recorded using a "Trace" that is stored in the passive file system.

The circularity test is a "commissioning tool" function and can also be selected in the Commissioning area of the HMI operator interface. For further information, refer to Section "Circularity test".

5.6.2 Conventional friction compensation

5.6.2.1 Conventional friction compensation

Type of friction compensation

Conventional friction compensation is selected by entering the value 1 in machine data

MD32490 $MA_FRICT_COMP_MODE (friction compensation method).

Amplitude adaptation

In many cases, the injected amplitude of the friction compensation value does not remain constant over the whole acceleration range. For example, for optimum compensation with high accelerations, a smaller compensation value must be injected than for smaller accelerations. For this reason, friction compensation **with adapted injection amplitude** can be activated in cases where high accuracy is required (see diagram below). The function is activated for specific axes with machine data

MD32510 $MA_FRICT_COMP_ADAPT_ENABLE = 1

(friction compensation adaptation active).

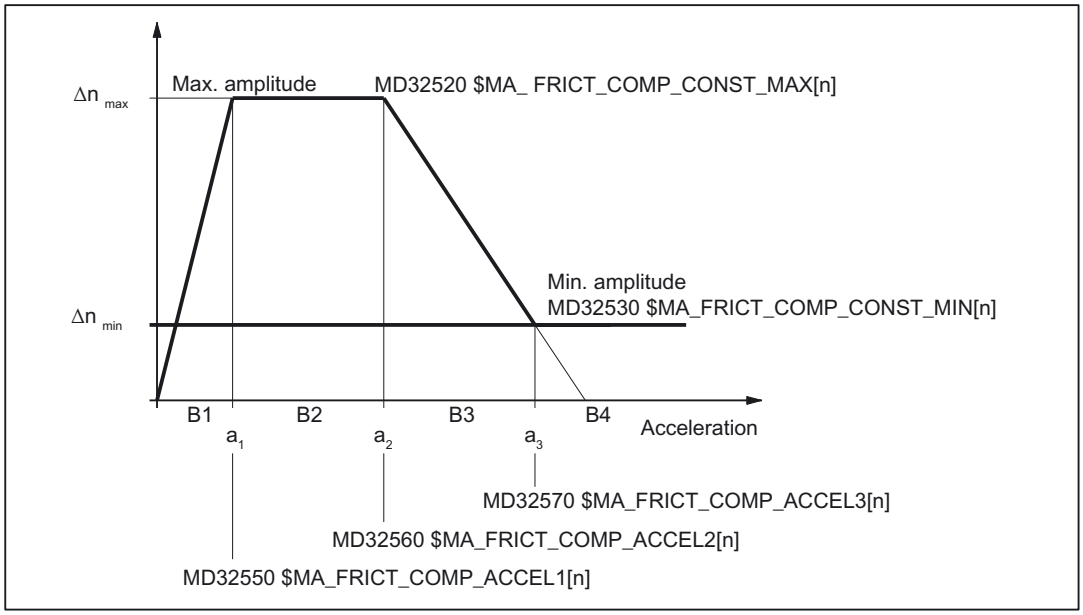


Figure 5-13 Typical curve for friction compensation with amplitude adaptation

The adaptation characteristic is divided into four ranges (a different injection amplitude Δn is applied in each range):

B1:	for $a < a_1$	$\Delta n = \Delta n_{max} * a / a_1$
B2:	for $a_1 \leq a \leq a_2$	$\Delta n = \Delta n_{max}$
B3:	for $a_2 < a < a_3$	$\Delta n = \Delta n_{max} * (1 - (a - a_2) / (a_3 - a_2))$
B4:	for $a \geq a_3$	$\Delta n = \Delta n_{min}$

Characteristic parameters

The parameters of the adaptation characteristic in the diagram above must be entered as machine data for specific axes.

Δn	=	Injection amplitude of friction compensation value
Δn_{max}	=	Maximum friction compensation value MD32520 $FRICT_COMP_CONST_MAX[n] (maximum friction compensation value)
Δn_{min}	=	Minimum friction compensation value MD32530 $FRICT_COMP_CONST_MIN[n] (Minimum friction compensation value)
a_1	=	Adaptation acceleration value 1 for friction compensation MD32550 $FRICT_COMP_ACCEL1[n] (adaptation acceleration value 1)

a_2 = Adaptation acceleration value 2 for friction compensation
MD32560 $FRICT_COMP_ACCEL2[n]
(adaptation acceleration value 2)

a_3 = Adaptation acceleration value 3 for friction compensation
MD32570 $FRICT_COMP_ACCEL3[n]
(adaptation acceleration value 3)

Note about characteristic shape

In exceptional cases, the calculation characteristic may deviate from the typical shape shown in the diagram above.

The value for Δn_{min} (MD32530) might then even be higher than Δn_{max} (MD32520).

5.6.2.2 Commissioning of conventional friction compensation

Circularity test

The friction compensation function can be commissioned most easily by means of a circularity test. Here, deviations from the programmed radius (especially at the quadrant transitions) can be measured and displayed while traversing a circular contour.

Step-by-step commissioning

The conventional friction compensation function must first be selected.

(MD32490 $MA_FRICT_COMP_MODE=1 (friction compensation method)).

The friction compensation value mainly depends on the machine configuration. Commissioning is performed in two stages.

- Stage 1: Calculation of the compensation values without adaptation
- Stage 2: Calculation of the adaptation characteristic (if the friction compensation is dependent on the acceleration and the results of stage 1 are not satisfactory).

Commissioning stage 1: Friction compensation without adaptation

1. Circularity test without friction compensation

A circularity test without friction compensation must be performed first

(MD32500 $MA_FRICT_COMP_ENABLE = 0)

The circularity test procedure is described in Section "Circularity test".

A typical characteristic of quadrant transitions without friction compensation is shown in the diagram below.

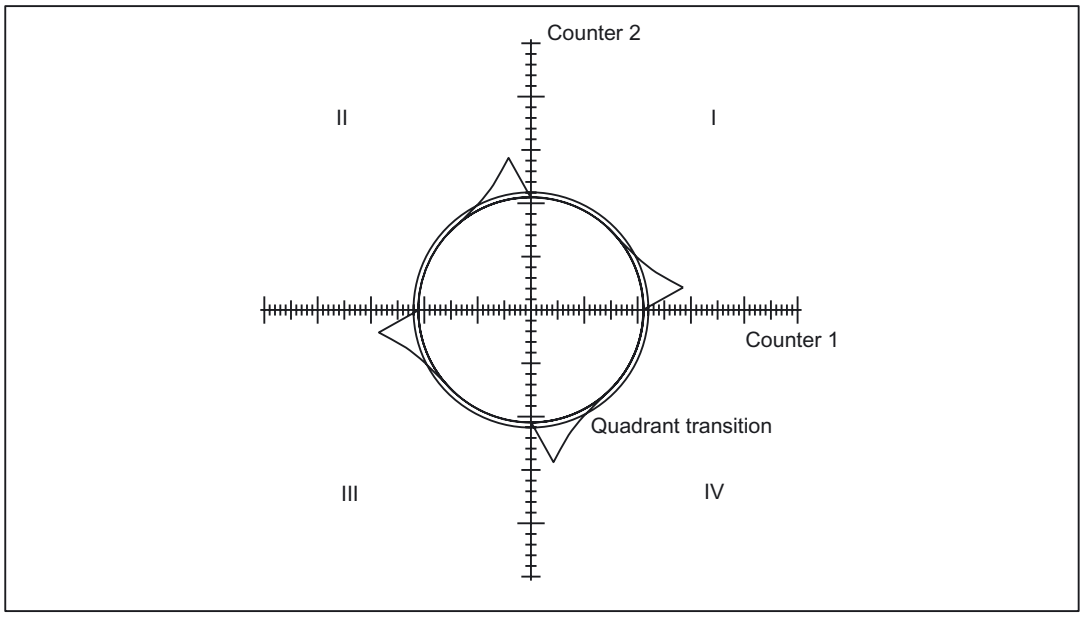


Figure 5-14 Uncompensated radius deviation at quadrant transitions

2. Enabling friction compensation

After this, the friction compensation must be activated for the axis/spindle in question.

Activate friction compensation with machine data

→ MD32500 $MA_FRICT_COMP_ENABLE[n] = 1

(friction compensation active)

3. Deactivate adaptation

In order to commission friction compensation without adaptation, the adaptation must be deactivated.

Deactivate adaptation with machine data

→ MD32510 $MA_FRICT_COMP_ADAPT_ENABLE[n] = 0

(friction compensation adaptation active)

4. Determine compensation parameters

Friction compensation without adaptation is defined by the following parameters:

1. MD32520 $MA_FRICT_COMP_CONST_MAX[n] (maximum friction compensation value) (amplitude) in [mm/min]
2. MD32540 $MA_FRICT_COMP_TIME[n] (friction compensation time constant) in [s]

These two parameters are changed until the circularity test produces minimum or no deviations from the programmed radius at the quadrant transitions. The tests must be performed with different radii and velocities (typical values for the application of the machine).

Starting value

A relatively low injection amplitude plus a time constant of a few position controller cycles should be entered as the start value when measuring commences.

Example:

MD32520 $MA_FRICT_COMP_CONST_MAX[n] = 10 (mm/min)

MD32540 $FRICT_COMP_TIME[n] = 0.008 (8 ms)

The effect of changing the parameters must be checked using the measured and plotted circles.

Mean value calculation

If it is not possible to determine a common compensation time constant for the varying radii and velocities, then the average of the calculated time constants must be worked out.

Good friction compensation setting

When the friction compensation function is well set, quadrant transitions are no longer noticeable.

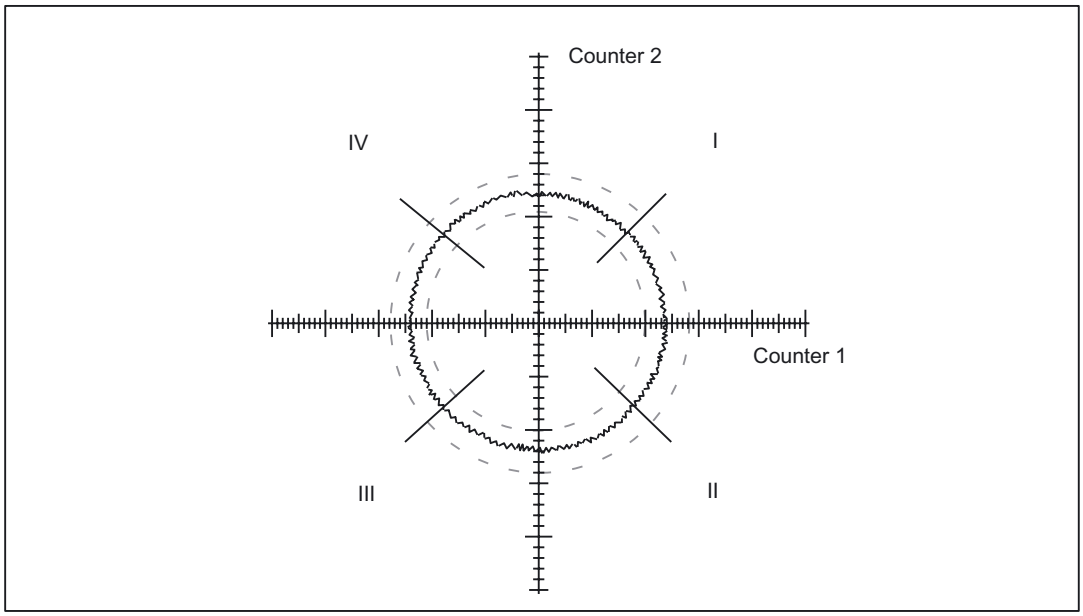


Figure 5-15 Quadrant transitions with correctly set friction compensation

Amplitude too low

When the injection amplitude setting is too low, radius deviations from the programmed radius are compensated inadequately at quadrant transitions during circularity testing.

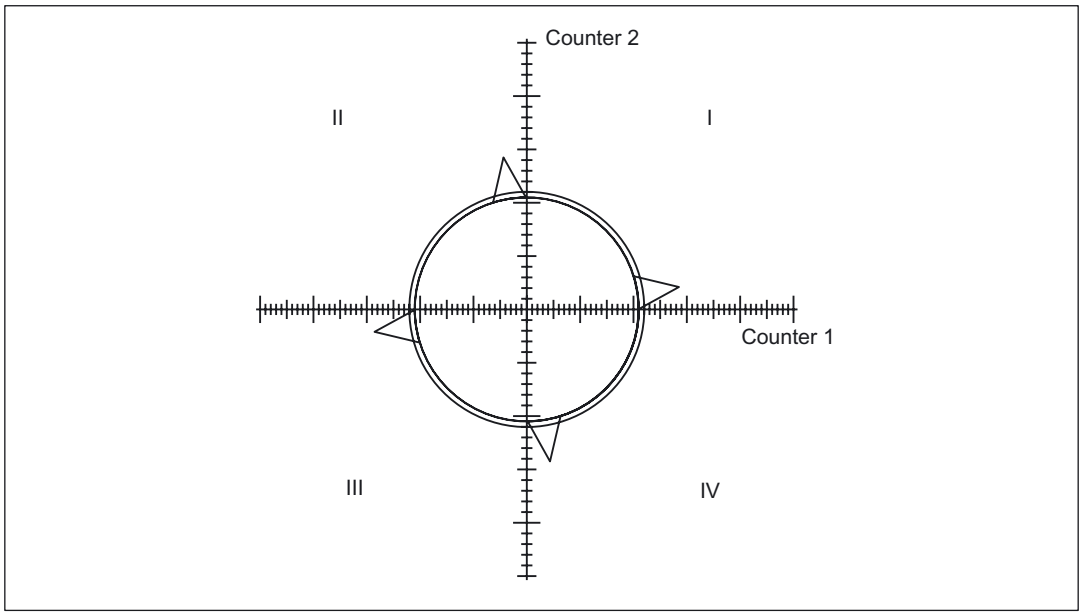


Figure 5-16 Amplitude too low

Amplitude too high

When the injection amplitude setting is too high, radius deviations at quadrant transitions are manifestly overcompensated at quadrant transitions.

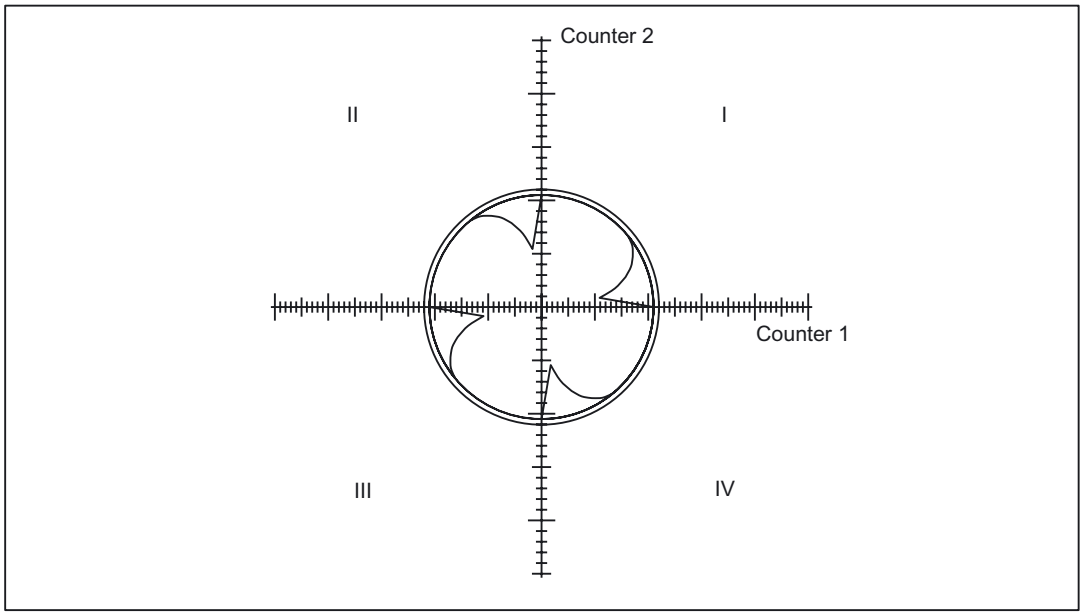


Figure 5-17 Amplitude too high

Time constant too low

When the compensation time constant settings are too low, radius deviations are compensated briefly at quadrant transitions during circularity testing, but are followed immediately again by greater radius deviations from the programmed radius.

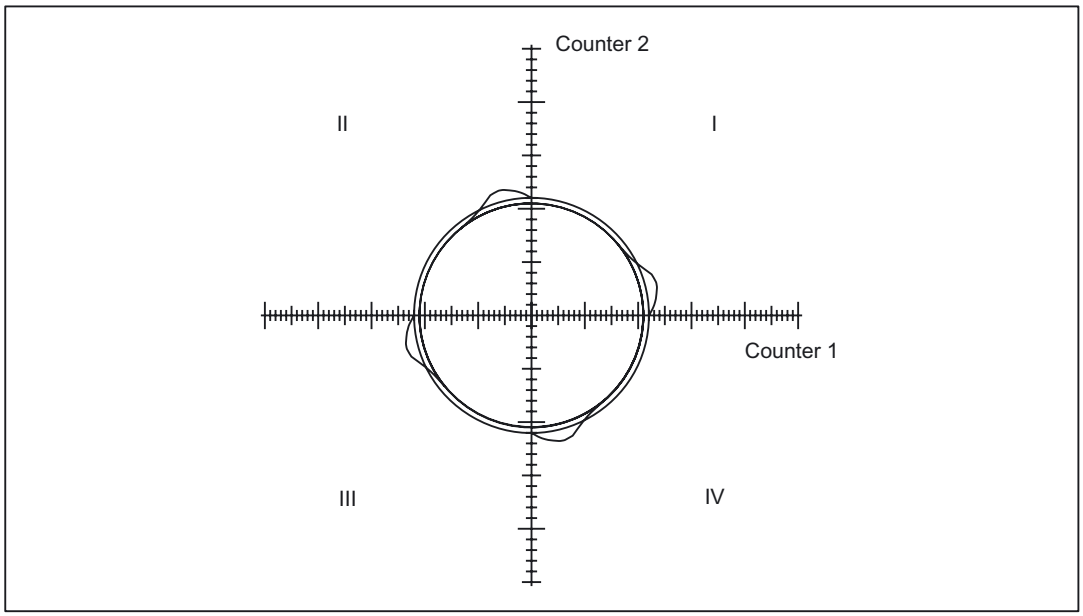


Figure 5-18 Compensation time constant too small

Time constant too high

When the compensation time constant settings are too high, radius deviations are compensated at quadrant transitions during circularity testing (on condition that the optimum injection amplitude has already been calculated), but the deviation in the direction of the arc center increases significantly after quadrant transitions.

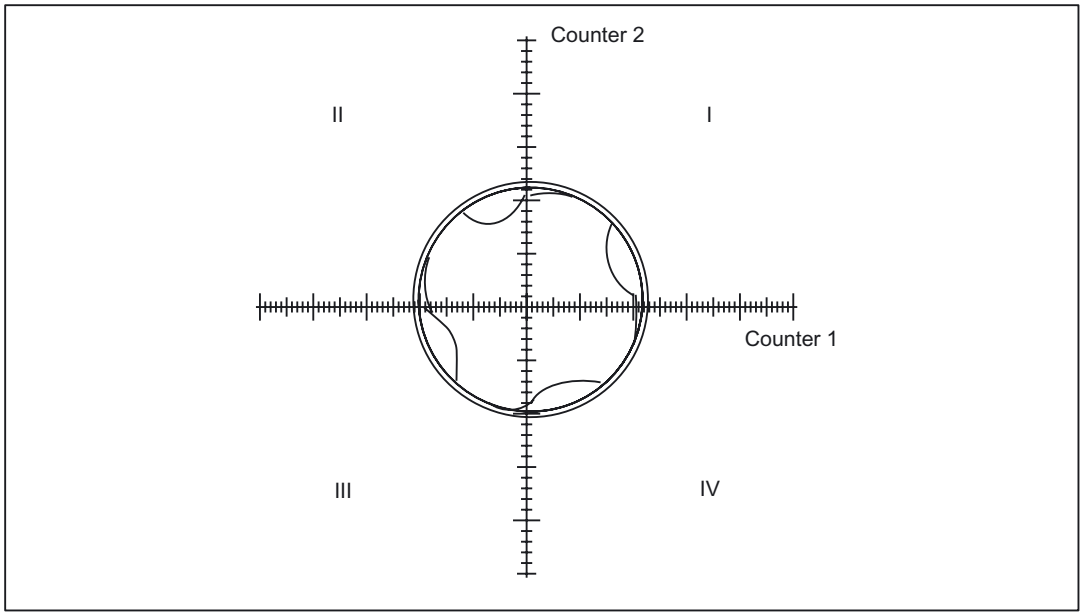


Figure 5-19 Compensation time constant too large

Adaptation yes/no?

If, with the time constant and the constant injection amplitude determined, a good result is achieved both in the circularity test and in positioning over the whole working area (i.e. for all radii and velocities of relevance), curve adaptation will not be necessary.

However, if the friction compensation turns out to be dependent on the acceleration, the adaptation characteristic must be calculated in second stage.

Commissioning stage 2: Friction compensation with adaptation

Application

Whenever friction compensation depends on the acceleration and the required results cannot be obtained with constant injection amplitude, adaptation must be used.

In order to obtain optimum compensation over the whole of the working range of the friction feedforward control where high demands are made on accuracy, the acceleration dependency of the compensation value must be calculated. To achieve this, the dependency must be measured at various points in the working range between acceleration zero and the maximum planned acceleration. The adaptation characteristic derived from the measurement results is then entered in the above machine data axis-specifically.

1. Calculate the adaptation characteristic

For different radii and velocities ...

1. ... it is necessary to determine the required injection amplitudes,
2. ... it is necessary to check the compensatory effect of the injection amplitudes using the circularity test,
3. ... it is necessary to log the optimum amplitudes.

The adaptation characteristic is defined completely by the settings of the parameters specified in Section "Conventional friction compensation". However, many more measured values must be obtained for checking purposes. It must be ensured that there is a sufficiently large number of interpolation points for small radii at high speed. The size of the curves must be obtained by plotting.

2. Determining acceleration values

For circular movements, the axial acceleration values are calculated with radius r and the traversal velocity v according to the formula

$a = v^2/r$

The velocity and thus also the axial acceleration value a can be varied easily by means of the feedrate override switch.

The acceleration values a_1, a_2 and a_3 for the adaptation characteristic must be entered in the relevant machine data in compliance with condition $a_1 < a_2 < a_3$:

MD32550 $MA_FRICT_COMP_ACCEL1(adaptation acceleration value 1)

to

MD32570 MA_FRICT_COMP_ACCEL3 (adaptation acceleration value 3)

If the curve is wrongly parameterized, the alarm 26001 "Parameterization error for friction compensation" is output.

Example of characteristic settings

1. Calculate the existing acceleration

   The axial acceleration rate is calculated at the zero speed crossing of a circular movement with formula $a = v^2/r$.

   With a radius of r = 10 mm and a circular velocity of v = 1 m/min (=16.7 mm/s), the acceleration rate is thus $a = 27.8\ mm/s^2$.

2. Entry of characteristic break points

   The following acceleration rates have been calculated as characteristic break points:

   $a_1 = 1.1\ mm/s^2$; $a_2 = 27.8\ mm/s^2$; $a_3 = 695\ mm/s^2$

The following values are therefore entered in the machine data in this order:

MD32550 $MA_FRICT_COMP_ACCEL1 [n] (adaptation acceleration value 1) = 0.0011 [m/s^2]

MD32560 $MA_FRICT_COMP_ACCEL2 [n] (adaptation acceleration value 2) = 0.0278 [m/s^2]

MD32570 $MA_FRICT_COMP_ACCEL3 [n] (adaptation acceleration value 3) = 0.695 [m/s^2]

For example, the following values were calculated for the injection amplitudes:

MD32520 $FRICT_COMP_CONST_MAX [n] = 30 [mm/min]

MD32530 $FRICT_COMP_CONST_MIN [n] = 10 [mm/min]

Note

If the results obtained at very low velocities are not satisfactory, then the computational resolution for linear positions must be increased in machine data:

MD10200 $MA_INT_INCR_ PER_MM (computational resolution for linear positions)

or for angular positions in machine data:

MD10210 $MA_INT_INCR_PER_DEG (computational resultion for angular positions)

See also machine data

MD32580 $MA_FRICT_COMP_INC_FACTOR

(weighting factor of friction compensation value with short traversing movements)

5.6.3 Quadrant error compensation using neural networks - only 840D sl

5.6.3.1 Fundamentals

Principle of QEC

The purpose of quadrant error compensation (QEC) is to reduce contour errors occurring during reversal as the result of drift, backlash or torsion. Compensation is effected through prompt injection of an additional speed setpoint.

In conventional QEC, the intensity of the compensation pulse can be set according to a characteristic as a function of the acceleration. This characteristic must be determined and parameterized during commissioning using the circularity test. The procedure for this is relatively complicated and requires some experience.

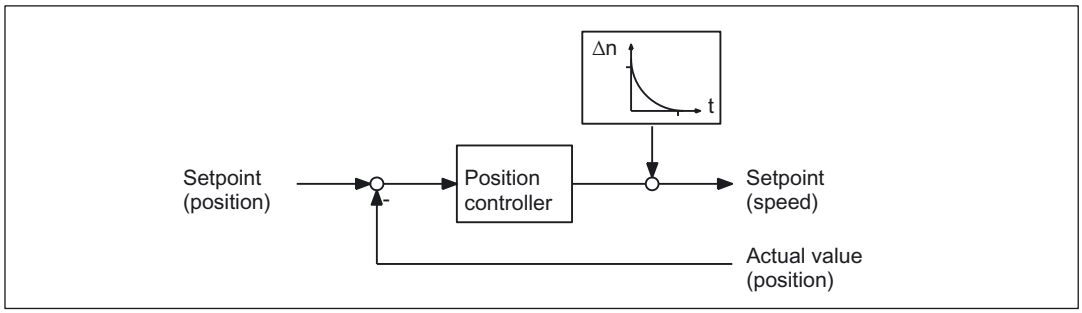


Figure 5-20 Injection of an additional speed setpoint pulse

Advantages of QEC with neural network

On the SINUMERIK 840D, the characteristic block that used to be manually parameterized can now be replaced by a neural network. This has the following advantages:

- Commissioning has been simplified because the compensation characteristic no longer needs to be set manually by the commissioning engineer but is **determined automatically** during a learning phase.
- The characteristic for a manually parameterized friction compensation is approximated by a polygon with 4 straight lines. For improved precision, the neural network can **reproduce the real curve much better**.

  The resolution of the characteristic curve can be adapted to the precision requirements and a directional quality of the compensation amplitude can be considered.

  In addition to the compensation amplitude, it is possible to adapt the decay time to the acceleration in special cases.
- The system permits **simple automatic re-optimization on site** at any time.

Requirements for neural QEC

An essential requirement for implementing QEC with neural network is that the errors occurring on the workpiece at quadrant transitions are detected by the measuring system. This is only possible either with a direct measuring system, with an indirect measuring system with clear reactions of the load on the motor (i.e. rigid mechanics, little backlash) or with suitable compensation. With indirect measuring systems, any backlash that might occur must be compensated by backlash compensation.

Learning/working phases

QEC with neural network involves the following two phases:

- Learning phase

  A certain type of response is impressed upon the neural network during the learning phase. The relation between the input and output signals is learnt. The result is the learnt compensation characteristic that is stored in the non-volatile user memory. The learning operation

  is activated or deactivated from the NC part program with special high-level language commands.

- Working phase

  During the working phase, additional speed setpoint pulses are injected in accordance with the learnt characteristic. The stored characteristic does not change during this phase.

The learning phase can be executed for several (up to 4) axes at the same time. For further information about neural network learning, see Section "Learning the neural network".

The learning and working phases and the resulting neural QEC are purely axial. There is no mutual influence between the axes.

Saving characteristic values

On completion of the learning phase, the calculated compensation data (characteristic values in user memory) including the network parameters (QEC system variables) must be saved in a file selected by the operator. These files are named "AXn_QEC.INI" per default.

Loading characteristic values

These saved and learned compensation data can be loaded back directly to the user memory in the same way as part programs.

When the part program containing the tables is loaded, the compensation values are transferred to the NC user memory. The characteristic values become effective only after compensation has been enabled.

Characteristic values **cannot** be written when the compensation function **is active** (machine data

MD32500 $MA_FRICT_COMP_ENABLE (friction compensation active) must be set to 0

and must be active).

For the QEC function:

The QEC must be enabled (and activated) with the following machine data:

MD32500 $MA_FRICT_COMP_ENABLE = 1 (QEC active)

Recommended commissioning procedure

As mentioned above, the neural network integrated in the control automatically adapts the optimum compensation data during the learning phase.

The axis involved must perform reversals with acceleration values constant section by section. Before activation of the learning phase, the parameters of the neural network (QEC system variables) must be pre-assigned in accordance with the requirements.

In order to simplify commissioning as much as possible, NC programs are provided as reference examples.

As described in subsection "Commissioning the neural QEC", the commissioning engineer must first learn the characteristic for the axes using these reference examples and the recommended QEC parameter values and check the contour accuracy achieved using the circularity test (see Section "Circularity test"). If the results do not meet the requirements, re-optimization must be performed changing the parameters appropriately (see Section "Parameterization of neural QEC", "Learning the neural network" and "Further optimization and intervention options") (so-called "relearning").

5.6.3.2 Parameterization of neural QEC

Machine data

The basic configuring data for the neural QEC are stored as machine data.

- MD32490 $MA_FRICT_COMP_MODE (friction compensation method)

  (2 = neural QEC)
- MD32500 $MA_FRICT_COMP_ENABLE

  (friction compensation active)
- MD32580 $MA_FRICT_COMP_INC_FACTOR

  (weighting factor friction compensation value with short traversing movements)
- MD38010 $MA_MM_QEC_MAX_POINTS

  (maximum number of compensation values for QEC with neural networks)

With these machine data, the neural QEC is activated as soon as the memory space is reserved in the non-volatile RAM. The procedure and assignments are described in Section "Commissioning the neural QEC".

All other data are set using system variables.

QEC system variables

The data for parameterizing the neural network are defined as system variables that can be written and read by an NC program. The following system variables are used for parameterization of the neural network:

- $AA_QEC_COARSE_STEPS

  "Coarse quantization of characteristic"

  This parameter defined the coarse quantization of the input signal and is therefore the resolution of the characteristic. The larger the value that is selected, the higher the memory requirement and the greater the length of time required for the training phase. See the end of this section for more information.

  Range of values: 1 to 1024; recommended value: 49

- $AA_QEC_FINE_STEPS

  "Fine quantization of characteristic"

  This parameter defines the fine quantization of the input signal and is therefore the resolution of the characteristic. The larger the value that is selected, the higher the memory requirement.

  Range of values: 1 ... 16; recommended value: 8

- $AA_QEC_DIRECTIONAL

  "Directionality"

  This parameter defines whether the compensation is to be injected directionally or not. If activated, a separate characteristic is determined and stored for each acceleration direction. Because two characteristics are used, double the memory space must be reserved in the non-volatile user memory.

  Range of values: `TRUE/FALSE`; recommended value: `FALSE`

- $AA_QEC_LEARNING_RATE

  "Learning rate for active learning phase"

  With the learning rate it is possible to determine how quickly the optimum characteristic is to be learnt in the active learning phase of the neural QEC. This value is a weighting factor with which it is possible to define to what extent the deviations affect the injection amplitude. With higher values (>100%), the characteristic is learned more quickly but too high learning rate values (weighting factors) can cause instability (two-step response).

  A small learning rate is recommended for relearning processes during normal operation (< 50%) otherwise the characteristic is changed on every little disturbance when the speed passes through zero.

  Range of values: > 0%; ≤ 500%; recommended value: 50%

- $AA_QEC_ACCEL_1/_2/_3

  "Acceleration limit values for the characteristic areas 1/2/3"

  The acceleration characteristic is divided into three areas. In each area there is a different quantization of the acceleration steps. In the low acceleration range, an especially high resolution is required for the characteristic in order to reproduce the widely varying compensation values there. For this reason, the input signals are quantized more finely, the smaller the acceleration is.

  Recommended values for

  – $AA_QEC_ACCEL_1: 20 mm/s^2 (= 2% of $AA_QEC_ACCEL_3)
  – $AA_QEC_ACCEL_2: 600 mm/s^2 (= 60% of $AA_QEC_ACCEL_3)
  – $AA_QEC_ACCEL_3: 1000 mm/s^2(maximum acceleration of working range)

  The value of the parameter $AA_QEC_ACCEL_3 must be entered as appropriate to the requirements; i.e. the neural network only works and learns optimally in this range. If a higher acceleration is detected than the parameterized working area, the injection amplitude that was determined during the defined maximum acceleration of the working range is used. At high accelerations, this injection value is relatively constant.

  The recommended values must only be changed if the compensation is insufficient in these acceleration ranges. For further information, please refer to Section "Further optimization and intervention options".

- $AA_QEC_TIME_1

  "Time constant for the neural QEC (decay time)"

  With this, the decay time of the compensation setpoint pulse is set if adaptation of the decay time is not used.

  The optimum decay time must be ascertained manually using the circularity test at a working point in the mid acceleration range. The procedure is described in detail in connection with conventional friction compensation (Section "Commissioning of conventional friction compensation") (analogous to machine data

  MD32540 $FRICT_COMP_TIME (friction compensation time constant)).

  With the recommended value (15 ms), it is possible to achieve good results.

  Range of values: ≥ 0; recommended value: 0.015s

  If the decay time adaptation is active, then $AA_QEC_TIME_1 determines the filter time constant in the center of the operating range (i.e. with 0.5 * $AA_QEC_ACCEL_3).

- **$AA_QEC_TIME_2**

  "Compensation time constant for adaptation of compensation value decay time"

  At a value of zero of less than or equal to $AA_QEC_TIME_1, no adaptation is performed.

  The decay time is usually constant over the entire working range. In rare cases however, it can be advantageous to raise the decay time in the very small acceleration range, or to lower it at high accelerations. For further information, please refer to Section "Further optimization and intervention options".

  Range of values: ≥ 0; recommended value: 0.015s (identical to $AA_QEC_TIME_1)

- **$AA_QEC_MEAS_TIME_1/_2/_3**

  "Measurement time for calculating the error criterion in acceleration range 1/2/3"

  The measurement time is started, as soon as the criterion for injection of the compensation value is fulfilled (i.e. the set speed changes sign). The end of the measurement time is defined by the set parameter values.

  Different measuring times are required for each characteristic range.

  Recommended values for

  – $AA_QEC_MEAS_TIME_1: 0.090s (= 6 *$AA_QEC_TIME_1)
  – $AA_QEC_MEAS_TIME_2: 0.045s (= 3 * $AA_QEC_TIME_1)
  – $AA_QEC_MEAS_TIME_3: 0.030s (= 2 * $AA_QEC_TIME_1)

  The recommended values must only be changed if the compensation is insufficient in these acceleration ranges or if $AA_QEC_TIME_1 is changed. For further information, please refer to Section "Further optimization and intervention options".

Parameter acceptance

The QEC system variables are stored in the non-volatile user memory after the NC program is started where they remain unchanged until the memory is erased or reformatted or until a new learning or relearning process takes place or until they are written by the NC program.

Before the learning cycle is called, all system variables must be assigned valid values for the learning process. This can be done, for example, in a subroutine. After this NC program has run and a reset has been performed, the QEC data are active.

Characteristic data

The characteristic data determined during the learning process are stored as system variables in the user memory reserved for this purpose.

Format: **$AA_QEC[n]** Range of n: 0 ... 1024

These values write the learned characteristic in internal formats and must therefore **not be changed**!

Quantization of characteristic

The quantization, and thus the resolution, of the characteristic is defined via the two quantities **fine quantization** ($AA_QEC_FINE_STEPS) and **coarse quantization** ($AA_QEC_COARSE_STEPS). The finer the resolution, the higher the memory requirement and the longer the duration of time required for the learning phase.

The number of memory locations required and the total number of quantization intervals is calculated by the formula:

Number of memory locations = $AA_QEC_FINE_STEPS * ($AA_QEC_COARSE_STEPS+1)

Up to 1025 memory locations per axis can be reserved. In this way, a sufficiently high resolution is achieved for high precision requirements.

The following 3 diagrams illustrate the meaning of the characteristic values for coarse and fine quantization, and their effect on the teach-in period, as a function of the parameter "Detailed learning active y/n". Three cases are distinguished for better understanding.

Case 1:

Coarse quantization > 1; fine quantization = 1 (special case; usually the fine quantization is in the region of eight):

In this case, the interpolation points of the characteristic are determined solely by coarse quantization (see diagram below).

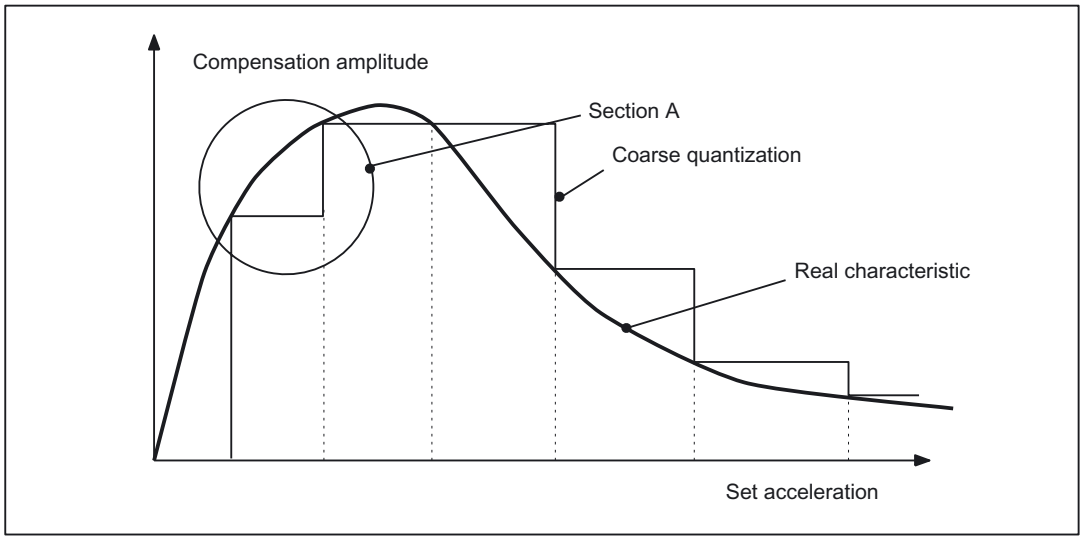


Figure 5-21 Coarse quantization of characteristic

Case 2:

Coarse quantization > 1; fine quantization > 1; "Detailed learning" is deactivated (this setting is the default):

In this case, discrete linear interpolation is used for fine quantization between the interpolation points of the coarse quantization.

The learning duration is identical with 1 because learning only occurs at the interpolation points of the coarse quantization.

The effect of fine quantization on a section of characteristic within a coarse quantization process is shown in the diagram below (see also Section A in diagram above).

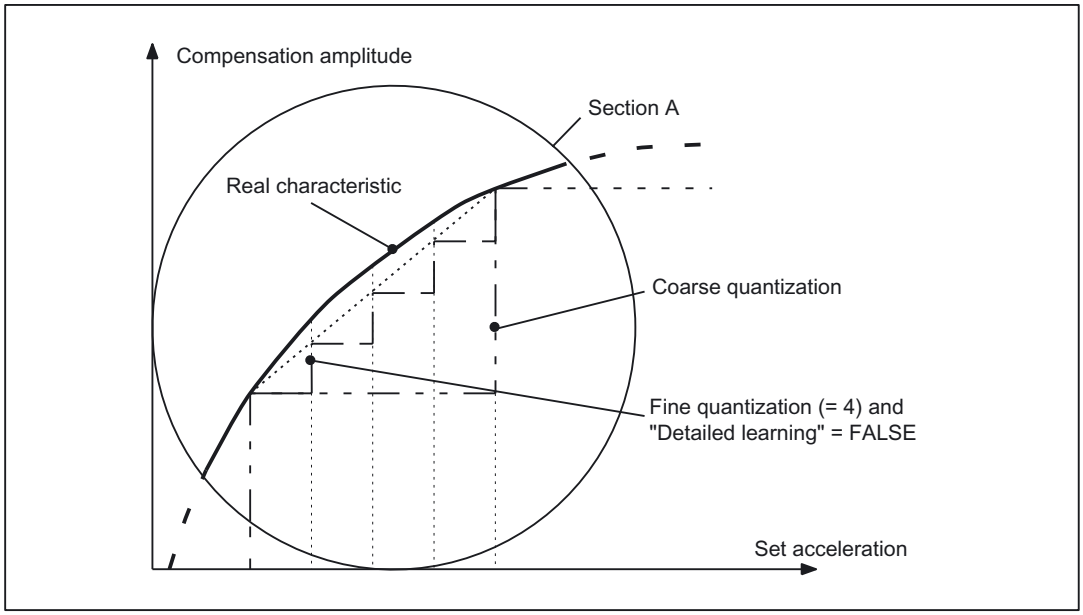


Figure 5-22 Effect of fine quantization with "Detailed learning" inactive

Case 3:

Coarse quantization > 1; fine quantization > 1; "Detailed learning" active (its use is only recommended for very high precision requirements):

With "Detailed learning", learning occurs both at the interpolation points of the coarse quantization and of the fine quantization.

The learning duration is therefore much longer.

The diagram below shows a severely fluctuating characteristic curve on which the effect of selecting and deselecting the "Detailed learning" function is clear.

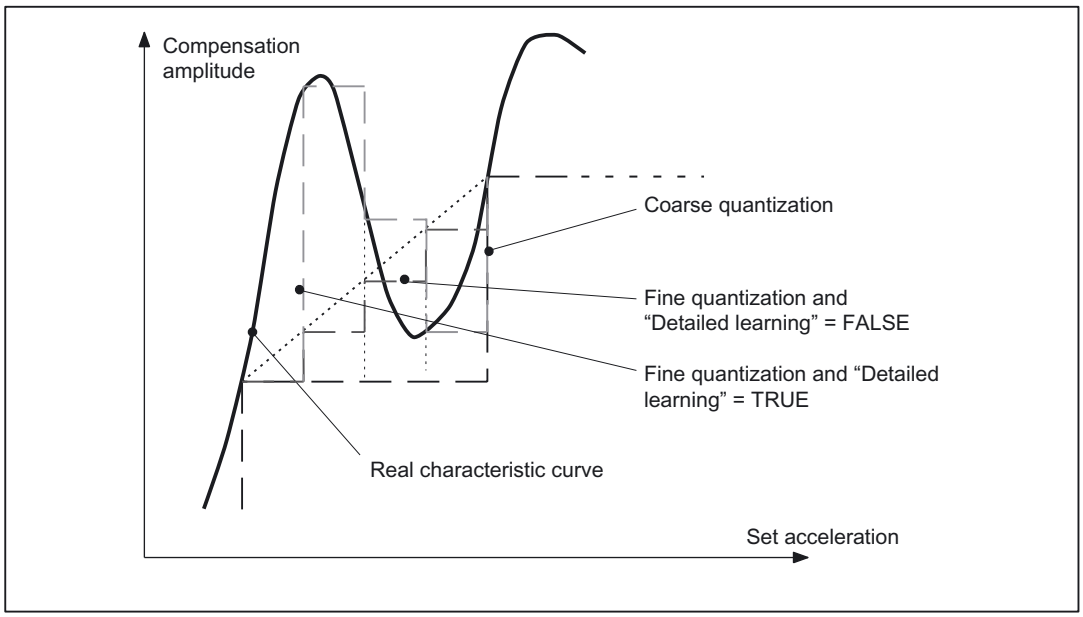


Figure 5-23 Effect of fine quantization with "Detailed learning" = active

5.6.3.3 Learning the neural network

Learning process sequence

A certain type of response is impressed upon the neural network during the learning phase. The relation between the input and output signals is learnt.

The learning process is controlled entirely by NC programs and is divided into the following areas:

1. Preset the QEC system variables for the learning process
2. Activate QEC system variables (by starting the NC program)
3. Parameterize the learning cycle
4. Start the learning cycle

The result is the learnt compensation characteristic that is stored in the non-volatile user memory.

The results achieved must be checked using the circularity test (Section "Circularity test").

Reference NC programs

In order to ease the task of the engineer in commissioning the QEC with neural networks, NC programs containing specimen routines for learning movements and assignments of QEC system variables (recommended values) are available.

These are the following reference NC programs:

- QECLRNP.SPF

  Learning with POLY standards (Option "POLY" necessary)

- QECLRNC.SPF

  Learning with circles

- QECDAT.MPF

  Sample NC program for assigning system variables and the parameters for the learning cycle

- QECSTART.MPF

  Reference NC program that calls the learning cycle

These NC programs are contained on the diskette of the basic PLC program for the SINUMERIK 840 D.

Implementing the learning process solely via NC programs has the following advantages:

- Learning can be fully automatic without operator intervention. This is advantageous for series commissioning operations if the optimum learning parameters for a machine type have been found and only the characteristic for each individual machine remains to determined or retrained.
- Learning can take place simultaneously for several axes (up to 4). This significantly reduces the learning phase for the machine.
- The traverse movements can easily be adapted to special requirements.

Note

The circularity test is an integral component of HMI Advanced. The commissioning tool must be used with HMI Embedded.

Learning motion

The axis traversing motions that must be executed to learn a specific response are generated by an NC program. Each learning motion of the sample learning cycle comprises a group of NC blocks with parabolic movements (ensuring that the axis traverses at the most constant possible setpoint speed after the zero crossing; see diagram below) in which the axes oscillate at constant acceleration in each program section. The acceleration is decreased from group to group. In the diagram below, NC blocks 2 to 3, 5 to 6 and 8 to 9 each form a group; the transitional movements to lower acceleration rates are programmed in blocks 1, 4, 7 and 10.

Note

So that the learning parameters act as preset, the feedrate override switch must be set to 100% during the learning phase.

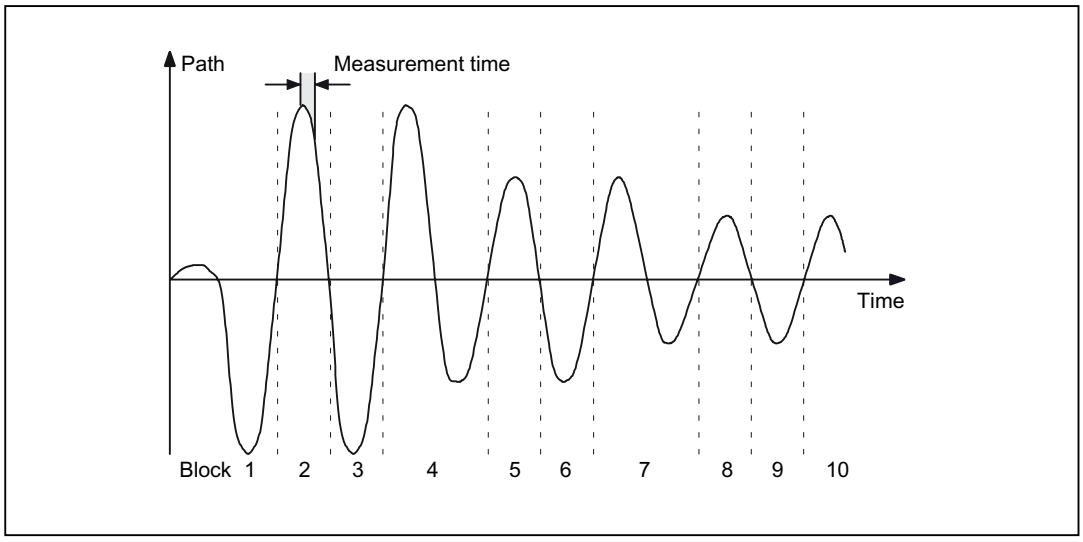


Figure 5-24 Typical traverse motion of an axis when learning the QEC characteristic

Assignment of system variables

Before a learning cycle is called, all QEC system variables must be set to the values required for the learning process. The values recommended in the reference NC program must be checked and changed if necessary (see Section "Parameterization of neural QEC").

Learning ON/OFF

The actual learning process of the neural network is then activated in the reference NC program. This is done using the following high-level language command:

QECLRNON(axis name 1, ... 4) Learning ON (for specified axes)

Only during this phase are the characteristics changed.

After the learning motions of the required axes have been completed, the learning process is deactivated for all axes. This is done with the high-level language command

QECLRNOF Deactivate learning (simultaneously for all axes)

After power-on reset, end of program (M02/M30) or operator panel front reset, learning is also deactivated.

The current "Learning on/off" status is displayed in the service display "Axes" with "QEC learning active" (1 = active; 0 = inactive).

Learning cycle call

Once learning has been activated, the reference NC program calls the learning cycle by means of the following input parameters:

- **Number of axes** to which learning is to apply (up to four).

  Prerequisites:

  If more than one axis is to learn at the same time, all QEC system variables of the axes involved must have the same values. These values are monitored and an error message is output if they are not equal.

- **Names of the learning axes**

  Initial number (same for all axes) Value always 0 (setpoint branch)

  Learning mode (initial learning = 0; relearning = 1)0: Initial learning active. All values of the network are preset to 0 before learning.

  1: Relearning active. Learning continues with the values already learnt in the defined step width.

- **Detailed learning active** yes/no (`TRUE`/`FALSE`)

  FALSE: "Detailed learning" is not active. The characteristic is therefore learnt in the step width of the coarse quantization of the acceleration.

  TRUE: "Detailed learning" is active. The characteristic is therefore learnt in the step width of the fine quantization of the acceleration, i.e. with fine quantization of 10 steps per coarse step, determination of the characteristic takes ten times longer. This parameter must therefore only be used for extremely high precision requirements.

Note

If "Detailed learning" is selected, the number of learning passes can and must be reduced in order to reduce the learning duration (recommended range: between 1 and 5).

- **Number of learning passes**

  Default value = 15; range > 0

  The effect of this parameter depends on whether "Detailed learning active" is set or not.

  a) Detailed learning not active (= FALSE):

  The number of test motions (back and forth) is defined for each acceleration stage. The higher the number, the more accurate learning is, but the longer learning takes.

  With directional compensation ($AA_QEC_DIRECTION = TRUE), the parameterized number of test movements for every direction is generated.

  b) Detailed learning not active (= TRUE):

  The number of complete runs, from maximum to minimum acceleration and vice versa, is activated with fine incrementation, i.e. with a setting of 1, all acceleration stages are executed once starting at the maximum value. For every acceleration stage, two test movements are generated if there is no directional compensation ($AA_QEC_DIRECTION = FALSE), otherwise four test movements are performed per acceleration stage.

  A reduction of the "Number of learning passes" can be made if data blocks for the machine type already exist (series machines) and these are to be used as a basis for further optimization.

- **Section-by-section learning active** yes/no (TRUE/FALSE)

  "Section-by-section learning" in certain acceleration ranges is especially interesting for "Detailed learning" e.g. in technologically important ranges of the machine. By defining the ranges appropriately it is possible to reduce the learning duration.

  Default value = FALSE

- **Range boundaries for "Section-by-section learning"** (minimum acceleration, maximum acceleration); only relevant for "Section-by-section learning active".

  Default value = 0; format: mm/s^2

- **Time taken for one test motion** (to and fro)

  Default value = 0.5; format: s (seconds) (corresponds to a frequency of 2 Hz)

Requirements

In the learning phase, the neural QEC requires a speed feedforward control without jerk limitation (BRISK):

(MD32620 $MA_FFW_MODE=1; FFWON (feedforward control mode))

The feedforward control must therefore be correctly parameterized and optimized. When the learning process is started a check is made to see whether the speed feedforward control is activated. If not, the learning process is canceled and an error message is generated.

5.6.3.4 Commissioning of neural QEC

General information

Commissioning the QEC function with neural networks is described in brief below. As we have already mentioned, the compensation characteristics during the learning phase are determined automatically.

The axis involved must perform reversals with acceleration values constant section by section. The QEC system variables for parameterization of the neural network must also be preset to meet the requirements.

To simplify commissioning as much as possible, NC programs are provided to serve as reference examples (see Section "Learning the neural network").

In the learning process, a distinction is made between "initial learning" (especially for first commissioning) and "relearning" (especially for re-optimization of characteristics already learnt). The procedures of "initial learning" and "relearning" are described below.

If the compensation characteristics for the machine are to be learned for the first time, we recommend use of the reference NC programs specified in Section "Learning the neural network".

"Initial learning" sequence

"Initial learning” -> cycle parameters "Learning mode” = 0

1. a) Activate QEC with neural networks for the required axes with machine data setting:

   MD32490 $MA_FRICT_COMP_MODE = 2 (friction compensation method)

   Note

   QEC with neural networks is an option!

   b) Reserve memory space for the compensation points with machine data

   MD38010 $MA_MM_QEC_MAX_POINTS

   (number of values for quadrant-error compensation with neural network)

   If the required number is not yet known, a generous amount of memory must be reserved initially (see also item 12).

   c) Parameterize and optimize the speed feedforward control (required for the learning phase)

   d) Perform a hardware reset (because of the re-allocation of the non-volatile user memory)

2. Activate QEC system variables:

   Adapt the reference NC program QECDAT.MPF for assigning the QEC system variables for all axes concerned (if necessary use the recommended values) and start the NC program. If error messages are output, correct the values and restart the NC program.

3. Create the NC program that moves the machine axes to the required positions and parameterizes and calls the reference learning cycle QECLRN.SPF (as in the example program QECSTART.MPF). The feedrate override switch must be set to 100% of the learning phase so that the parameters can take effect in accordance with the defaults.

4. Activate the learning phase by starting this NC program. The compensation characteristic is learned simultaneously for all parameterized axes. The learning duration depends on the specified learning parameters. If default values are used, it can take several minutes. The status of the axes concerned can be observed in the service display "axis" in the display "QEC learning active".
5. Activation of the injection of the compensation values for the required axes with machine data setting:

   MD32500 $MA_FRICT_COMP_ENABLE = 1(friction compensation active).
6. Parameterize the trace for the circularity test in the menu "Circularity test measurement" (with HMI Advanced or commissioning tool). Parameter values for reference NC program:

   Radius[mm]:

   Feedrate[mm/min].

   After this, enable the measuring function with the vertical softkey "Start".
7. Start the NC program with test motion (circle). The actual position values during the circular movement are recorded and stored in the passive file system. After termination of data recording, the recorded contour is displayed as a diagram.
8. Check the quadrant transitions for the contour recorded.
9. Depending on the result, repeat items 2, 4, 7, 8 and 9 if necessary. It might be necessary to change certain QEC system variables first (see also Section "Learning the neural network").
10. The compensation characteristics must be saved as soon as the contour precision meets the requirements (see Section "Learning the neural network").
11. If necessary, the memory area previously reserved for the compensation values can be reduced to the memory actually required.

NOTICE
When the machine data below is altered, the non-volatile user memory is automatically re-allocated on system power-on. MD38010 $MA_MM_QEC_MAX_POINTS (number of values for quadrant-error compensation with neural network) All the user data in the non-volatile user memory are lost. These data must therefore be backed up first. After power-on of the control, the backed up characteristics must be loaded again.

"Relearning"sequence

"Relearning” -> cycle parameters "Learning mode" = 1

"Relearning" can be used to perform a simple, automatic re-optimization process on previously learned characteristics. The values already in the user memory are taken as the basis.

The reference NC programs adapted to the machine (e.g. from "initial learning") must be used in the learning phase for "relearning". Generally, the previous values of the QEC system variables can still be used. Before the learning cycle is called, the parameter "learning mode" must be set to 1 (meaning "relearning"). It might also be used to reduce the "number of training passes".

Sequence of operations for "Relearning"

The sequence of operations involved in the Relearning process is described below.

1. If characteristic values have not yet been stored in the user memory (RAM) (e.g. commissioning of a series machine), the pre-optimized data block must be loaded (see Section "Fundamentals").
2. Adapt the NC program that moves the machine axes to the required positions and parameterizes and calls the learning cycle. The parameters for the learning cycle (e.g. QECLRN.SPF) might have to be changed for "relearning".
   - Set "Learn mode" = 1
   - Reduce the "number of learning passes" if necessary (e.g. to 5)
   - Activate "section-by-section learning" if necessary and define the associated range boundaries
3. Activate the learning phase by starting this NC program. The compensation characteristic is learned simultaneously for all parameterized axes.
4. Parameterize the trace for the circularity test in the menu "Circularity test measurement" (with HMI Advanced or commissioning tool). After this, enable the measuring function with the vertical softkey "Start".
5. Start the NC program with test motion for circularity test. The actual position values during the circular movement are recorded and stored in the passive file system. After termination of data recording, the recorded contour is displayed on the operator interface.
6. Check the quadrant transitions for the contour recorded.
7. Depending on the result, repeat items 3, 4, 5 and 6 if necessary. It might be necessary to change certain QEC system variables first (see also Section "Further optimization and intervention options").
8. The compensation characteristics must be saved as soon as the contour precision meets requirements (see Section "Fundamentals").

5.6.3.5 Further optimization and intervention options

Optimization options

In cases where the results of the circularity test do not meet the required accuracy standards, the system can be further improved by selective changes to QEC system variables. Several ways of optimizing the neural QEC are explained here.

Alteration of coarse and fine quantization

The input value is quantized by the two variables "coarse quantization" and "fine quantization".

A high value for the fine quantization causes a "similar" output signal to be obtained for adjacent intervals of the input signal, allowing, for example, measuring errors which occur only at a particular acceleration rate to be identified.

With a low fine quantization, highly fluctuating characteristics are reproduced better.

For the neural friction compensation, it is necessary to make use of the largest error tolerance by setting a high fine quantization ($AA_QEC_FINE_STEPS in the region of 5 to 10).

Directional compensation

Direction-dependent friction compensation must be used in cases where compensation is not applied equally on opposing quadrants when compensation values are being injected independently of direction (see diagram below).

The directional injection is activated via the system variable $AA_QEC_DIRECTIONAL = TRUE.

Here, the following aspects must be observed:

- Since a characteristic is learned and stored for every direction of acceleration, double the memory space is required in the non-volatile user memory. The machine data below must be adjusted accordingly.

  MD38010 $MA_MM_QEC_MAX_POINTS

  (number of values for quadrant-error compensation with neural network)

- The number of learning passes must be raised because only every second passage occurs at the same location.
- If the characteristic resolution is the same, commissioning takes longer.

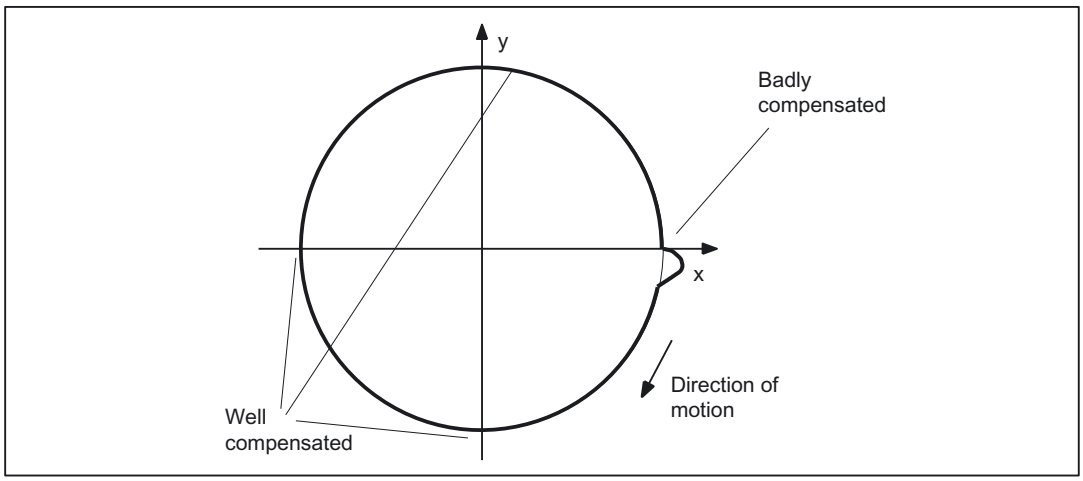


Figure 5-25 Example of directional friction compensation (circularity test)

Changing the characteristic ranges

The acceleration characteristic is divided into three ranges. In the low acceleration range, an especially high resolution is required for the characteristic in order to reproduce the widely varying compensation values there. Therefore, the lower the acceleration rate, the finer the quantization of the input quantity (see diagram below).

In the high acceleration range, there are only small changes in the compensation values so that a small resolution is perfectly sufficient.

The percentage settings recommended in Section "Parameterization of neural QEC" for $AA_QEC_ACCEL_1 (2% of $AA_QEC_ACCEL_3) and for $AA_QEC_ACCEL_2 (60% of $AA_QEC_ACCEL_3) are based on empirical values measured on machines with a maximum acceleration rate (= operating range) of up to approx. 1 m/s^2.

If the working range is significantly reduced, then the limit values for a_1 and a_2 must be set somewhat higher as a percentage of a_3. However, $AA_QEC_ACCEL_1 must not exceed the range of approx. 5% of the maximum acceleration. Useful boundaries for $AA_QEC_ACCEL_2 are approx. the values 40% to 75% of the maximum acceleration.

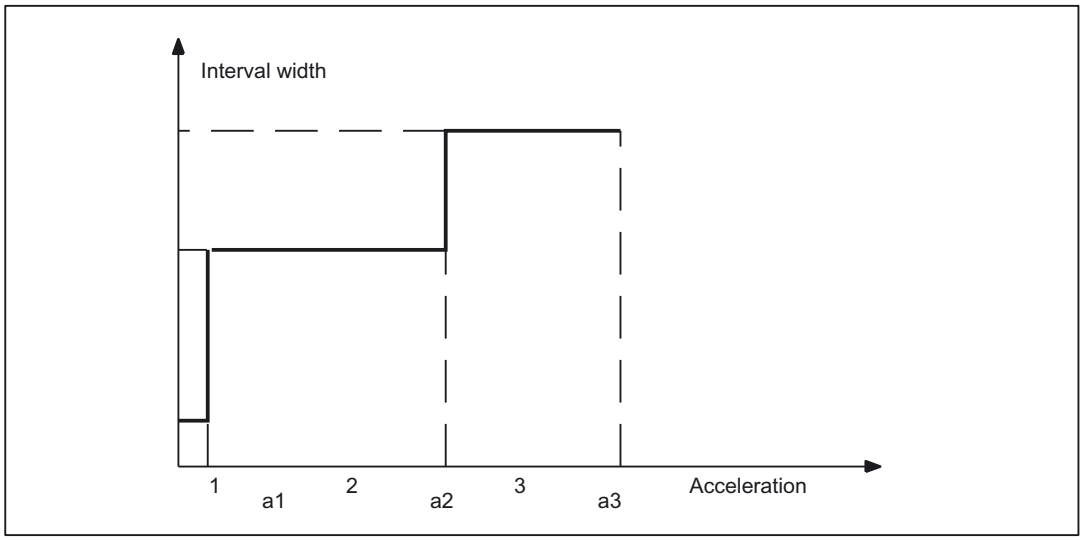


Figure 5-26 Interval width in acceleration ranges

Adaptation of the decay time

In special cases, it is possible to adapt the decay time of the compensation setpoint pulse in addition to the compensation amplitude.

If, for example, the circularity test reveals that in the low acceleration range (a_1) the quadrant transitions yield good compensation results but that radius deviations occur again immediately after this, it is possible to achieve an improvement by adapting the decay time.

The time constant without adaptation ($AA_QEC_TIME_1) is only valid in the mid acceleration range (50%).

The adaptation of the decay time for the compensation setpoint impulse according to the characteristic shown in the diagram below is parameterized with system variable $AA_QEC_TIME_2 (for acceleration = 0). The adaptation is formed by these two points according to an e^{-x} function (see diagram below).

The adaptation is performed under the following condition: $AA_QEC_TIME_2 > $AA_QEC_TIME_1

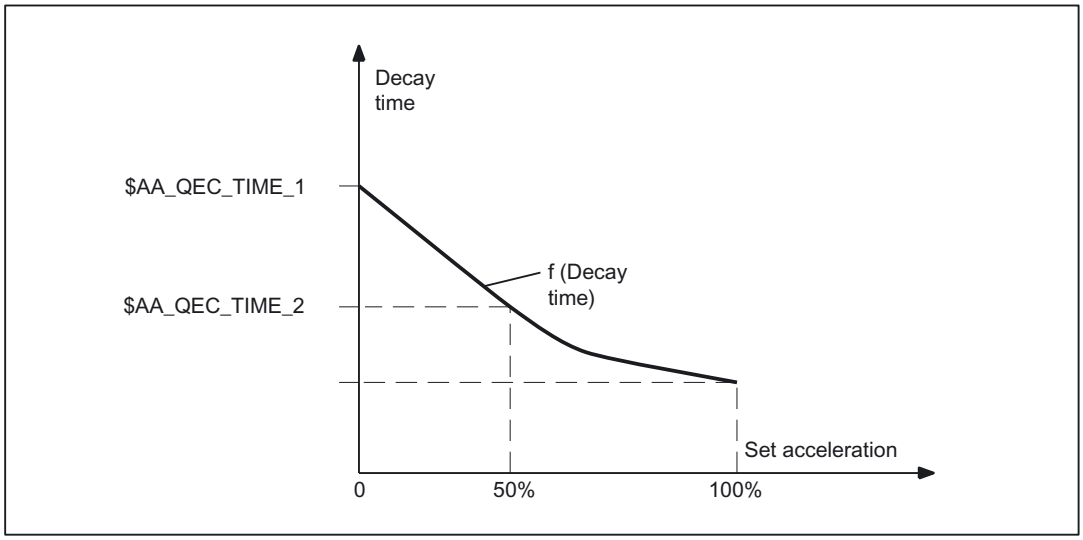


Figure 5-27 Adaptation of the decay time

Alteration of error measuring time

During the learning phase for the neural network, the error measuring time determines the time window within which contour errors are monitored after a zero-speed passage.

Experience has shown that the error measuring time to be used for average acceleration rates (approx. 2 to 50 mm/s^2) corresponds to three times the value of the decay time ($AA_QEC_MEAS_TIME_2 = 3 * $AA_QEC_ TIME_1).

In the very low and high acceleration ranges, the error measuring time must be adapted. This is done automatically according to the characteristic in the diagram below. The error measuring time for small accelerations is set to six times the value of the decay time ($AA_QEC_MEAS_TIME_1 = 6 * $AA_QEC_ TIME_1); double the decay time ($AA_QEC_MEAS_TIME_3 = 2 * $AA_QEC_ TIME_1) is taken as the error measuring time for larger accelerations.

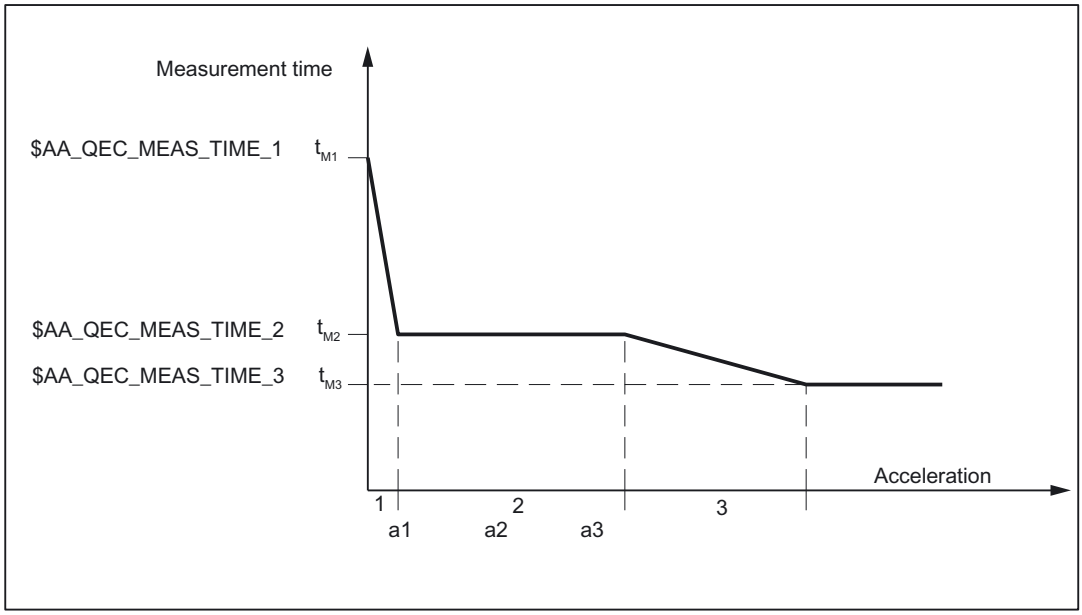


Figure 5-28 Dependency of error measuring time on acceleration rate

In special cases, it might be necessary to reparameterize the error measuring times:

- Setting of very extreme values for the QEC compensation time constant. Experience indicates that it is not useful to set an error measuring time of less than 10ms or more than 200ms.
- Parameterization of the error measuring times with adaptation of the decay time of the compensation value

  If the adaptation of the decay time of the compensation value is active (see above), the following rule of thumb is applicable to the parameterization of the error measuring time for acceleration range 1:

  $AA_QEC_MEAS_TIME_1 = 3 * $AA_QEC_ TIME_2

  Example:

  Decay time ($AA_QEC_TIME_1) = 10ms

  Adaptation of decay time ($AA_QEC_TIME_2) = 30ms

  The above rule of thumb produces an error measuring time for acceleration range 1 of:

  $AA_QEC_MEAS_TIME_1 = 3 * 30ms = 90ms

  Without decay time adaptation, the calculated setting for $AA_QEC_MEAS_TIME_1 would be merely 6 * 10ms = 60ms.

Overcompensation with short traversing motions

Practical experience has shown that the optimum friction compensation value calculated from the circularity test may result in overcompensation on the relevant axis if it executes very short axial positioning movements (e.g. on infeeds in the mm range).

To improve accuracy in such cases too, it is possible to reduce the compensation amplitude for short traversing motions.

MD32580 $MA_FRICT_COMP_INC_FACTOR

(weighting factor friction compensation value with short traversing movements)

This weighting factor specified in the above machine data automatically takes effect when friction compensation is activated (conventional QEC or QEC with neural networks) acting on all positioning movements that are made within an interpolation cycle of the control.

The input range is between 0 and 100% of the calculated compensation value.

Control of learning process duration

As described in previous sections, the duration of the learning process is dependent on several parameters. It is mainly dependent on the following values:

- Coarse quantization ($AA_QEC_COARSE_STEPS)
- Measuring time for determining the error criterion ($AA_QEC_MEAS_TIME_1 up to $AA_QEC_MEAS_TIME_3)
- Number of learning passes
- Detailed learning active [yes/no]?
- Fine quantization ($AA_QEC_FINE_STEPS) (only if "detailed learning active = yes" is selected)
- Directional compensation active [yes/no]? ($AA_QEC_DIRECTIONAL)
- Duration of reversing movement

The setting “Detailed learning active = yes” causes a significant increase in the time required for learning. It must therefore only be used where precision requirements are high. It is necessary to check whether these requirements only apply to certain acceleration ranges. If so, detailed learning only needs to be performed section by section (see "Section-by-section learning y/n?"). The number of learning passes must be reduced in any case.

If the reference NC programs mentioned above are used with the recommended parameter values, the following times have been determined for the learning process time:

- Detailed learning not active: approx. 6.5 min
- Detailed learning active: approx. 13 min

5.6.3.6 Quick commissioning

Preparation for "Learning"

- Calculate the optimum friction compensation time constant

  (MD32540 $MA_FRICT_COMP_TIME (backlash)) with the conventional friction compensation.

- **Enter the following machine data without power ON:**

Machine data	Standard	Change to	Meaning
MD19330 NC-CODE_CONF_NAME_TAB[8]	0		Activate option "IPO_FUNKTION_MASK". **Only with learn program "Polynomial"! Bit4 = 1**
MD19300 COMP_MASK	0		Set option
MD32490 $MA_FRIC_COMP_MODE (friction compensation method)	1	2	"Type of friction compensation" neural QEC
MD32500 $MA_FRIC_COMP_ENABLE (friction compensation active)	0	0	"Friction compensation active" for learning **"OFF"**
MD32580 $MA_FRIC_COMP_INC_FACTOR (weighting factor friction compensation value with short traversing movements)	0	0	"Weighting factor of friction compensation value for short traversing motions" (mm increments)
MD38010 $MA_MM_QEC_MAX_POINTS (number of values for quadrant-error compensation with neural network)	0	400	"Selection of values for QEC" = $AA_QEC_FINE_STEPS * ($AA_QEC_COARSE_STEPS + 1)
MD32620 $MA_FFW_MODE (feedforward control mode)	1	1	Speed feedforward control
MD32610 $MA_VELO_FFW_WEIGHT (feedforward control factor for speed)	1	1	Injection 100%
MD32630 $MA_FFW_ACTIVATION_MODE (activate feedforward control from program)	1	0	Feedforward control ON continuously
MD32810 $MA_EQUIV_SPEEDCTRL_TIME (equivalent time constant speed control loop for feedforward control)	0.004	Initial value t_pos + n_setSm.*	Adjust equivalent time constant n control loop

* t_position ... position control cycle (=basic system cycle * factor for position control cycle), n_setSm. ... speed setpoint smoothing (MD1500 to 1521)

- **Read in the machine data because of the memory change (MD38010).**
  - **HMI Embedded:**

    Back up "Services" "Data OUT" "Commissioning data, NCK data" and, if applicable, "LEC, measuring system error, sag and angularity error compensation tables" via PCIN. Perform a `POWER ON` reset and then read in the backup data using PCIN and "Data IN". (= series commissioning).
  - **HMI Advanced:**

    Back up "SERIES COMM." and, if applicable, "LEC, measuring system error, sag and angularity error compensation tables". Perform a `POWER ON` Reset and read in the "COMM." archive (backed up data are loaded again).
- **Copy the Toolbox programs to the NC (with archive!)**

  QECDAT.MPF

  QECSTART.MPF

  QECLRNP.SPF (learn program "Polynomial") **or** QECLRNC.SPF (learn program "Circle") is stored as QECLRN.SPF on the NC!

  The learn program "Circle" should be used where possible for GEO axes, but only the learn program "Polynomial" is recommended for axes of any other type.
- **Adapt the following programs:**
  - **In part program QECDAT**

    adjust the friction compensation time constant if necessary (see point 1)

    `N1340 $AA_QEC_TIME_1[outNr,axNr] = 0.0xx`

    `N1040 def int numAxes =` ..... enter the number of axes to be learned

    `N1150 axisName[0] =` ..... enter the axis name of the 1st axis.

    `N1160 axisName[1] =` ..... enter the axis name of the 2nd axis.

    `N1170 axisName[2] =` ..... enter the axis name of the 3rd axis.

    `N1180 axisName[3] =` ..... enter the axis name of the 4th axis.

    (For the "Circle" learn program, AX1 .. AX8 or the machine or channel axis name can be entered as the axis name. In contrast, only the channel axis name may be used for the "Polynomial" learn program)
  - **In part program QECSTART**

    (For the "Circle" learn program, AX1 .. AX8 or the machine or channel axis name can be entered as the axis name. In contrast, only the channel axis name may be used for the "Polynomial" learn program)

    `N1080 def int numAxes =` ..... enter the number of axes to be learned.

    `N1310 axisName[0] =` ..... enter the axis name of the 1st axis.

    `N1320 axisName[1] =` ..... enter the axis name of the 2nd axis.

    `N1330 axisName[2] =` ..... enter the axis name of the 3rd axis.

    `N1340 axisName[3] =` ..... enter the axis name of the 4th axis.

Executing "Learning" process

Start the following programs

- Select and start QECDAT. System variables are assigned.
- Select QECSTART and override 100% and start. The learn program takes about 15 minutes to execute with a traversing motion of about 30 cm. If the message "REORG not possible" is displayed, it can be ignored. The message is displayed for about 10 seconds. It then disappears and the learning process continues with traversing motions.

Activate QEC

Machine data	Standard	Change to	Meaning
MD32500 $MA_FRIC_COMP_ENABLE (friction compensation active)	0	1	Switch on "Friction compensation active"

"Circularity test"

Use the "Circularity test" to check the result!

Save compensation data

Save compensation data (QEC data are not included in back-up with "SERIES COMM."):

HMI Embedded:

Save with PCIN under SERVICES\Data\Circle error compensation\All

HMI Advanced:

Save the file Quadrant_Error_Comp-complete in directory NC Active Data \ Quadrant error compensation under SERVICES. This file contains all compensation values.

Note

Change the "displayed name length" to "20" in SERVICES "System settings" "for display" to ensure that the whole name is visible.

5.7 Circularity test

Function

One of the purposes of the circularity test is to check the contour accuracy obtained by the friction compensation function (conventional or neural QEC). It works by measuring the actual positions during a circular movement and displaying the deviations from the programmed radius as a diagram (especially at the quadrant transitions).

Procedure

The circle contour for the axes involved is specified by an NC program. To simplify the circularity test as much as possible for the commissioning engineer, an NC program is provided as a reference example for the circularity test motion (file QECTEST.MPF on the diskette with the basic PLC program). The commissioning engineer must adapt this NC program to his application.

Several measurements must be made during the circularity test with different acceleration values to ascertain whether the learnt compensation characteristic (for neural QEC) or the defined compensation values (for conventional QEC) meet the requirements.

The circular movement can easily be made with different accelerations if you change the feedrate using the feedrate override switch without changing circular contour. The real feedrate must be taken into account in the measurement in the input field "feedrate".

The circle radius chosen must be typical of machining operations on the machine (e.g. radius in the range 10 to 200 mm).

For the duration of the circular movement, the position actual values of the axes are recorded and stored in a "trace" in the passive file system. The circularity test is therefore purely a measuring function.

Parameterization of circularity test

You select in this menu the **names** or numbers of the axes with which the circle is traversed and whose actual position data must be recorded. No check is made to find out whether the selected axes match the axes programmed in the NC part program.

The parameter settings in the input fields "Radius" and "Feed" must correspond to the values from the part program that controls the circular motion of the axes, taking account of the feed override switch setting. No check is made to see whether the values in the part program (including feedrate override) and the input values match.

The "Measuring time" display field shows the measuring time calculated from the "Radius" and "Feed" values for recording the position actual values during the circular movement.

If only parts of the circle can be represented (i.e. measuring time too short) the measuring time can be increased in the menu by reducing the feed value. This also applies if the circularity test is started from the stationary condition.

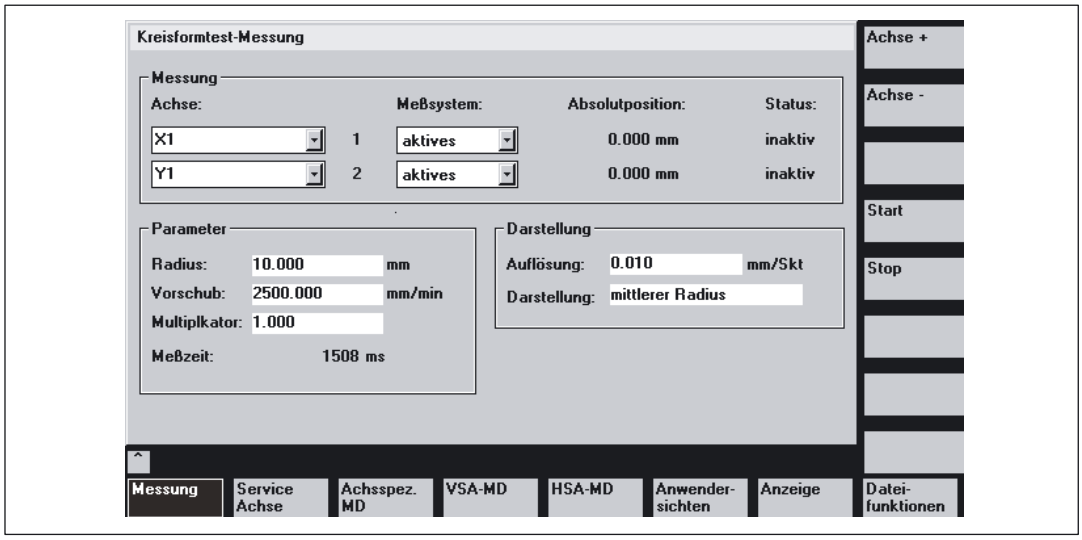


Figure 5-29 Circularity test measurement menu

Display mode

The following parameter assignments for programming the mode of representation of measurement results can also be made:

- Display based on mean radius
- Display based on programmed radius
- Scaling of the diagram axes

If the measuring time calculated exceeds the time range that can be displayed from the trace buffers (maximum measuring time = position control cycle frequency * 2048), a coarser sampling rate is used for recording (n * position control cycle frequency), so that a complete circle can be displayed.

Start measurement

The operator must use an NC Start to start the part program in which the circular motion for the selected axes is stored (`AUTOMATIC` or MDA operating mode).

The measuring function is started with the vertical softkey **Start**.

The sequence of operations (NC Start for part program and Start measurement) can be chosen by the user according to the application.

When the circularity test is active for the specified axes, the message "active" appears in the "Status" display field.

Stop measurement

The measurement can be stopped at any time by pressing the **Stop** softkey. Any incomplete measurement recordings are best displayed by selecting the **Display** softkey. There is no monitoring in this respect.

To allow direct access to the required controller parameters, the softkeys **Axis-specific MD, FDD-MD** and **MSD-MD** are displayed. The vertical softkeys **Axis+** and **Axis-** can be used to select the desired axis.

The "Service axis" display is displayed when you press the **Service Axis** softkey. The following service data are displayed here cyclically for commissioning of the friction torque compensation:

- QEC learning active yes/no?
- Current position and actual speed values

Display

When you press the **Display** softkey, the display switches to the graphical view of the recorded circle diagram.

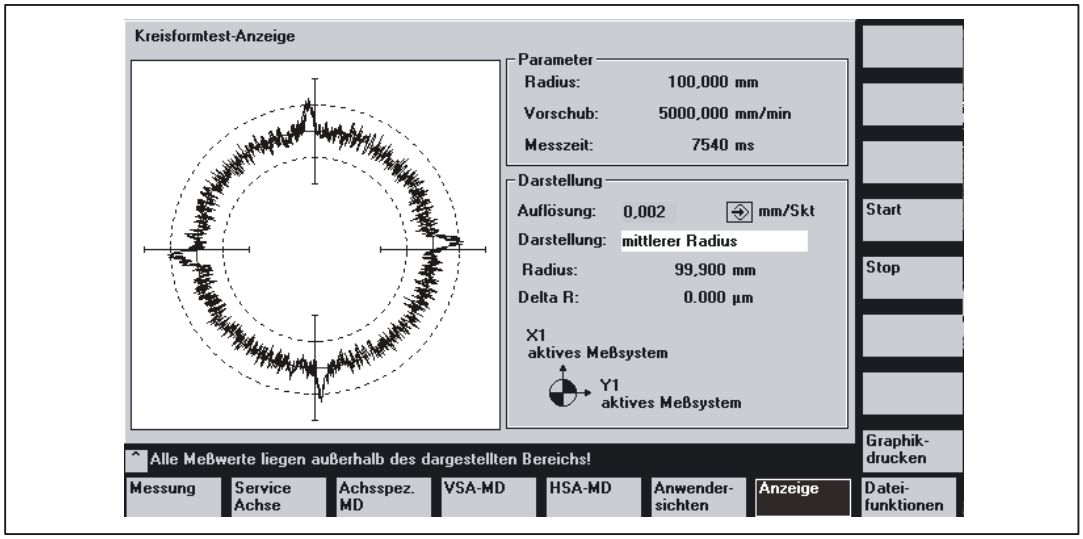


Figure 5-30 Circularity test display menu

This screen displays the measurements of the two actual position values as a circle with the set resolution.

The programmed radius, the programmed feedrate and the measuring time derived from these values are also displayed for documentation purposes (for subsequent storage of the measured circle characteristics in file format).

The operator can enter a finer scale for the diagram axes in the **Resolution** input field, e.g. in order to emphasize the transitions at the quadrants. The circle diagram is refreshed with the new resolution when you press the **Display** softkey.

File functions

The displayed measurement results and the parameter settings can be stored as a file on the MMC by selection of softkey **File Functions**.

Printer settings

The basic display for selecting a printer can be called by means of softkeys **HMI** \ **Printer selection**.

The toggle key is used to define whether the displayed graphic is to be output directly on the printer or transferred to a bit map file after softkey **Print graphic** is selected.

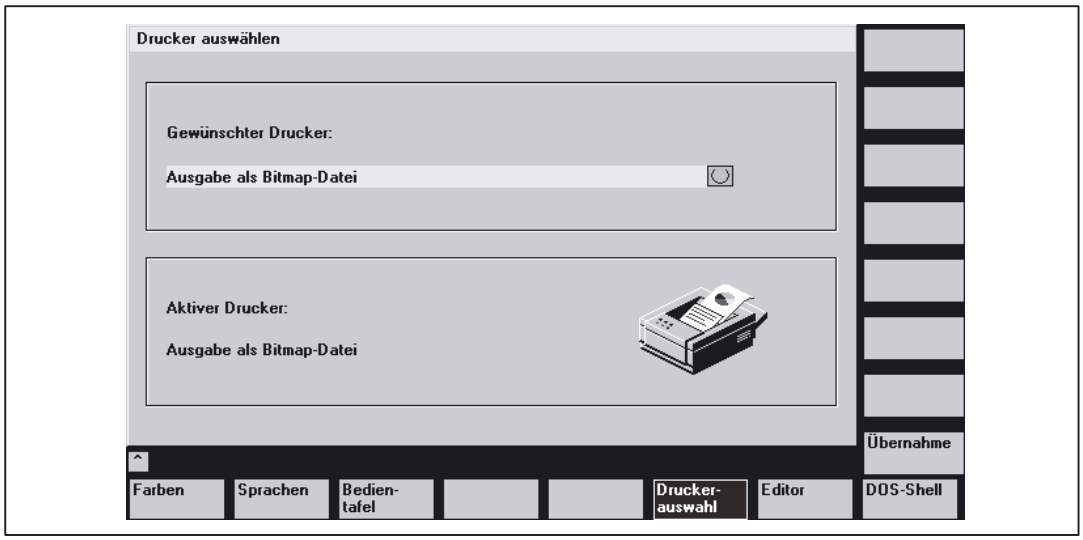


Figure 5-31 Basic screen for printer selection

Direct output on printer

The printer must be set up under MS-Windows.

"Output on printer" is selected in the dropdown menu. When softkey **Print graphic** is selected, the display graphic is output to the connected printer.

Output as bitmap file

The graphic is stored in a bitmap file (*.bmp).

"Output as bitmap file" is selected in the dropdown menu of printer settings.

The screen form for entering a file name is then displayed when softkey **Print graphic** is selected in the "Circularity test display" screen. A new file name can be entered or an existing file name selected for overwriting in the drop-down list.

The file is saved using the softkey **OK**. With the softkey **Cancel** you can return to the current graphic display.

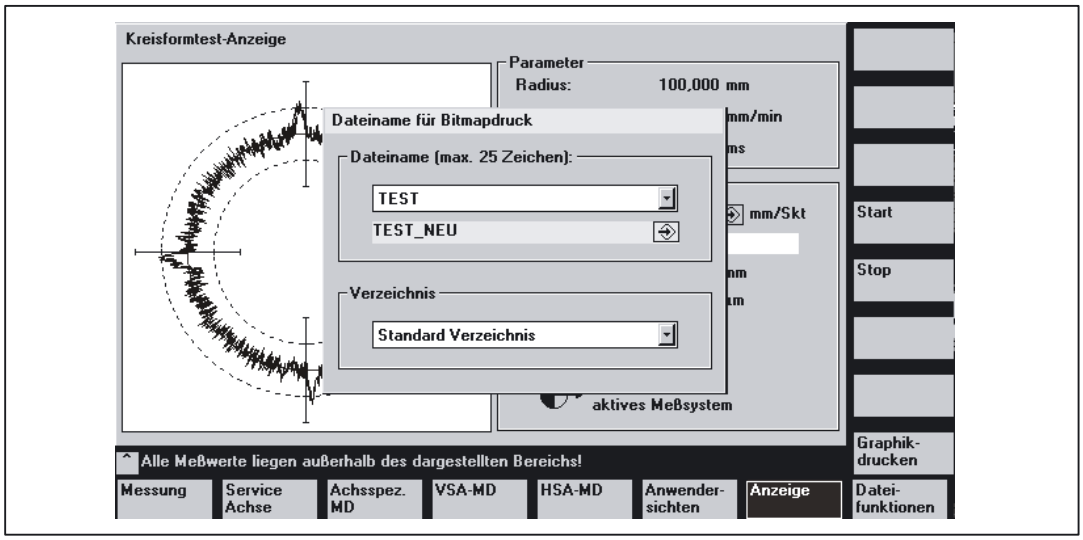


Figure 5-32 Assignment of file name for output in a bitmap file.

5.8 Measures for hanging (suspended axes)

5.8.1 Electronic counterweight

Axis without counterweight

For axes that have a weight load without counterweight, then after the brake is released, the hanging (suspended) axis drops and the following response is obtained:

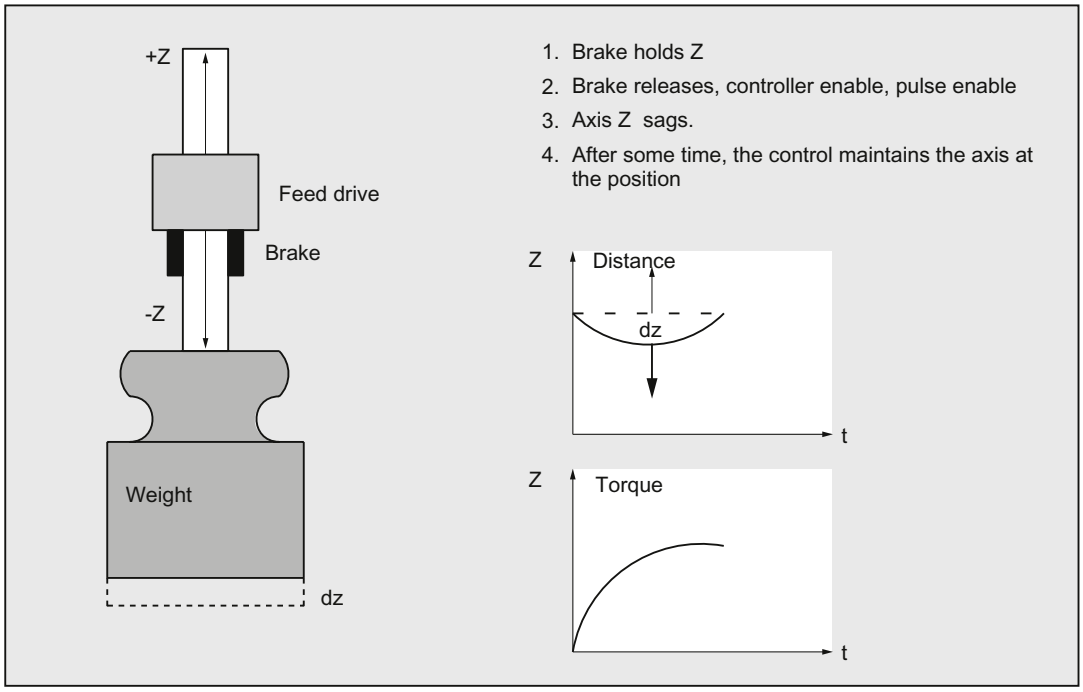


Figure 5-33 Drop of a hanging axis without counterweight

"Electronic counterweight" function

A hanging (suspended) axis can almost be completely prevented from dropping (sagging) using the "electronic counterweight" function.

The electronic counterweight prevents axes with a weight load from sagging when the closed-loop control is switched on. After releasing the brake, the constant counterweight torque maintains the position of the vertical axis.

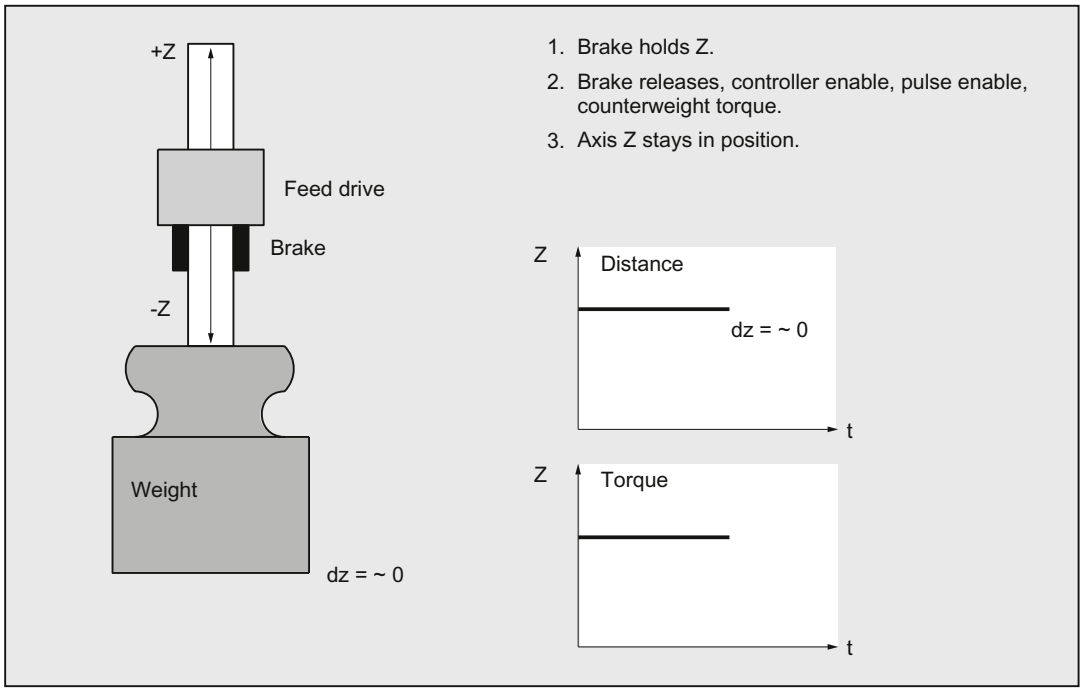


Figure 5-34 Lowering of a vertical axis with electronic weight compensation

Commissioning

Note

The "electronic counterweight" is commissioned through the drive!

Reference

For additional information, see the following:

SINAMICS S120 Function Manual Drive Functions

5.8.2 Reboot delay

Secondary effects of a reboot via HMI

HMI is capable of booting the NCK for the purpose, for example, of activating machine data. The result of this reboot would be that vertical axes would drop a short distance. The "reboot delay" function can be used to avoid this.

Reboot delay

The reboot delay results in the NCK and PLC being shut down with a delay and communicates the pending shutdown in order to prevent hanging (suspended) axes from dropping.

Note

The reboot delay only works with a controlled POWER ON via the HMI.

The reboot delay is not activated for a POWER FAIL (power failure) or a hardware reboot.

Reboot sequence

The HMI initiates an NCK and PLC reboot using PI service "_N_IBN_SS".

NCK immediately activates alarm 2900 in response to the PI service.

Mechanical axis brakes can be activated in the time that the NCK allows to expire from the PI service to the reboot (reboot delay time, refer to MD10088 $MN_REBOOT_DELAY_TIME).

Reactions to alarm 2900

- The following NC/PLC signals are **canceled**, i.e. set to zero:
  - DB11 DBX 6.3 (mode group ready) ; all mode groups
  - DB21, … DBX 36.5 (channel ready) ; all channels
  - DB31, … DBX 61.2 (axis ready) ; all axes
- The "Ready" message at relay contacts 72 73.1 73.2 74 is **not** reset.
- The motor brakes along the current limit.

  For further details, see machine data:
  - MD36610 $MA_AX_EMERGENCY_STOP_TIME (braking ramp time when errors occur)
  - MD36620 $MA_SERVO_DISABLE_DELAY_TIME (switch-off delay controller release)

  Note

  After the shutdown delay, the NCK withdraws the controller enable (MD36620) of the position control.

- The following NC/PLC interface signals remain at 1:

  DB10 DBX108.7 (NC ready)

By using the machine data:
MD11410 $MN_SUPPRESS_ALARM_MASK (mask for suppressing special alarms) (BIT20) the alarm 2900 is suppressed, however, the NCK triggers the same reactions.

As alarm 2900 deactivates the axis position control, this alarm must be configured to initiate that the **mechanical brakes are closed by the PLC**. Rebooting the PLC forces the PLC outputs to change to defined zero. The brakes must be connected up in such a way that they **remain closed at zero**, i.e. a 1 signal on the PLC allows the brakes to open.

Note

In terms of its reactions, the alarm is the same as the Emergency Stop alarm (3000). For internal reasons, the reboot delay time of the NCK can be slightly increased.

Activation

The reboot delay can be activated as follows:

MD10088 $MN_REBOOT_DELAY_TIME (reboot delay) **> 0**

The value that has been entered supplies the reboot delay time in seconds.

Evaluation with a system variable

System variable $AN_REBOOT_DELAY_TIME can be read in a synchronized action. A value greater than zero indicates that the reboot request has been initiated from the HMI and how much time (in seconds) the NCK plans until reboot (POWER OFF followed by POWER ON). In a synchronized action, the user can identify the pending reboot and appropriately respond (e.g. with "Safe Standstill" for a Safety Integrated application). $AN_REBOOT_DELAY_TIME is 0.0 as long as the HMI has not initiated a reboot request.

5.9 Data lists

5.9.1 Machine data

5.9.1.1 General machine data

Number	Identifier: $MN_	Description
10050	SYSCLOCK_CYCLE_TIME	Basic system clock cycle
10070	IPO_SYSCLOCK_TIME_RATIO	Factor for interpolator clock cycle
10082	CTRLOUT_LEAD_TIME	Shift of setpoint transfer time
10083	CTRLOUT_LEAD_TIME_MAX	Maximum permissible setting for shift of setpoint transfer time
10088	REBOOT_DELAY_TIME	Reboot delay
18342	MM_CEC_MAX_ POINTS[t]	Maximum number of interpolation points of sag compensation

5.9.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES	Reset G groups

5.9.1.3 Axis/Spindle-specific machine data

Number	Identifier: $MA_	Description
32450	BACKLASH	Backlash
32452	BACKLASH_FACTOR	Weighting factor for backlash
32490	FRICT_COMP_MODE	Type of friction compensation
32500	FRICT_COMP_ENABLE	Friction compensation active
32510	FRICT_COMP_ADAPT_ENABLE	Friction compensation adaptation active
32520	FRICT_COMP_CONST_MAX	Maximum friction compensation value
32530	FRICT_COMP_CONST_MIN	Minimum friction compensation value
32540	FRICT_COMP_TIME	Friction compensation time constant
32550	FRICT_COMP_ACCEL1	Adaptation acceleration value 1
32560	FRICT_COMP_ACCEL2	Adaptation acceleration value 2
32570	FRICT_COMP_ACCEL3	Adaptation acceleration value 3
32580	FRICT_COMP_INC_FACTOR	Weighting factor for friction compensation value for short traversing motion
32610	VELO_FFW_WEIGHT	Feedforward control factor for velocity/speed feedforward control

Number	Identifier: $MA_	Description
32620	FFW_MODE	Feedforward control mode
32630	FFW_ACTIVATION_MODE	Activate feedforward control from program
32650	AX_INERTIA	Inertia for torque feedforward control
32700	ENC_COMP_ENABLE	Interpolatory compensation
32710	CEC_ENABLE	Enabling of sag compensation
32711	CEC_SCALING_SYSTEM_METRIC	System of units for sag compensation
32720	CEC_MAX_SUM	Maximum compensation value for sag compensation
32730	CEC_MAX_VELO	Change of velocity during sag compensation
32750	TEMP_COMP_TYPE	Temperature compensation type
32760	COMP_ADD_VELO_FACTOR	Velocity increase as a result of compensation
32711	CEC_SCALING_SYSTEM_METRIC	System of units for sag compensation
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
32910	DYN_MATCH_TIME	Time constant for dynamic response adaptation
36500	ENC_CHANGE_TOL	Maximum tolerance for position actual value switchover
38000	MM_ENC_COMP_MAX_POINTS	Number of interpolation points with interpolatory compensation
38010	MM_QEC_MAX_POINTS	Number of values for quadrant-error compensation with neural network

5.9.2 Setting data

5.9.2.1 General setting data

Number	Identifier: $SN_	Description
41300	CEC_TABLE_ENABLE[t]	Enable evaluation of beam sag compensation table
41310	CEC_TABLE_WEIGHT[t]	Weighting factor for beam sag compensation table

5.9.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43900	TEMP_COMP_ABS_VALUE	Position-independent temperature compensation value
43910	TEMP_COMP_SLOPE	Gradient for position-dependent temperature compensation
43920	TEMP_COMP_REF_POSITION	Reference position for position-dependent temperature compensation

5.9.3 Signals

5.9.3.1 Signals from NC

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NC Ready	DB10.DBX108.7	DB2700.DBX2.7

5.9.3.2 Signals from mode group

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Mode group ready	DB11.DBX6.3	DB3100.DBX0.3

5.9.3.3 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Channel ready	DB21, ...DBX36.5	DB3300.DBX4.5

5.9.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1	DB31, ...DBX60.4	DB390x.DBX0.4
Referenced/synchronized 2	DB31, ...DBX60.5	DB390x.DBX0.5
Axis ready	DB31, ...DBX61.2	DB390x.DBX1.2

K5: Mode groups, channels, axis interchange

6

6.1 Brief description

Mode group

A mode group is a collection of machine axes, spindles and channels which are programmed to form a unit. In principle, a single mode group equates to an independent NC control (with several channels). A mode group is made up of those channels that always have to operate simultaneously in the same mode.

Note

There is one mode group as standard.

Reference:
Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

Note

Only 1 mode group is available for SINUMERIK 828D.

Channels

Every channel has its own program decoding, block preparation and interpolation functions. A part program can be processed independently within a channel.

Note

There is one channel available as standard.

Reference:
Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

The processes in several channels of a mode group can be synchronized in the parts programs.

Note

Only 1 channel is available for SINUMERIK 828D.

Axis/spindle interchange

After control system power ON, an axis/spindle is assigned to a specific channel and can only be utilized in the channel to which it is assigned.

With the function "Axis/spindle interchange" it is possible to enable an axis/spindle and to allocate it to another channel, that means to replace the axis/spindle.

Axis/spindle interchange can be activated via the parts program, via the PLC program and from motion-synchronous actions.

Axis/spindle interchange is also possible via:

- Programming in parts program GET/GETD.
- Automatically through programming of axis name.
- Without preprocessing stop and existing synchronization between preprocessing and main run.
- Through PLC via the VDI interface to the NCK.

Axis replacement extensions

- Set axis replacement behavior variable.
- Axis replacement with an axis container rotation with implicit GET/GETD.
- Axis replacement without pre-processing stop of axes not involved in the contour.
- Geometry axis with rotated frame (ROT) and axis replacement in JOG operational mode.
- Axis replacement via synchronized actions GET(axis), AXTOCHAN.

Note

For SINUMERIK 828D, an axis/spindle interchange is not possible between channels.

6.2 Mode groups - only 840D sl

Mode groups

A mode group combines NC channels with axes and spindles to form a machining unit.

A mode group contains the channels that are required to run simultaneously in the same mode from the point of view of the machining sequence.

Any axis can be programmed in any channel of a certain mode group. A mode group therefore corresponds to an independent, multiple-channel NC.

Example

On large machine tools (machining centers), it may be necessary for a parts program to be processed on one part of the machine while new workpieces to be machined need to be clamped and set up on another part. Such tasks usually require two independent NC controls.

With the mode group function, both tasks can be implemented on one NC control with two mode groups because a different mode can be set for each mode group (AUTOMATIC mode for the program processing, JOG for setting up a workpiece).

Mode group assignment

The configuration of a mode group defines the channels, geometry axes, machine axes and spindles which it is to contain.

A mode group consists of one or several channels which must not be assigned to any other mode group. Machine axes, geometry axes and special axes themselves are assigned to these channels. A machine axis can only be assigned to the channels of one mode group and can only traverse in this mode group.

A mode group is configured with the following data:

- Channel-specific machine data:

  MD10010 $MN_ASSIGN_CHAN_TO_MODE_GROUP (channel valid in mode group)

- Configuration data of the channels

Note

For more information about the first mode group, please refer to:

References:

/FB1/ Function Manual, Basic Functions; Mode Group, Channel, Program Operation Mode (K1)

6.3 Channels - only 840D sl

Note

The terms Channel, Channel Configuration, Channel States, Effects of Commands/Signals, etc. is described for the first channel in:
Reference:
Function Manual Basic Functions; Mode Group, Channel, Program Operation (K1)

For all other channels, this information applies, too.

6.3.1 Channel synchronization (program coordination)

General

As an example, double-slide machining operations or real-time processes can only be carried out if it is possible to synchronize processing in two channels. The channels affected shall perform certain processing procedures time-matched. To allow time-matched processing, the relevant channels must be joined to form a synchronization group (mode group).

The channel synchronization is programmed only via the NC language.

Prerequisite

The affected channels must be assigned to the **same mode group**.

Program coordination

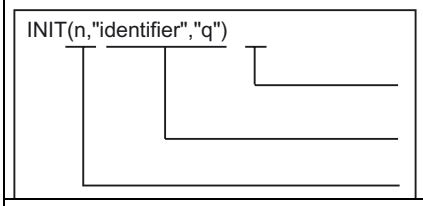
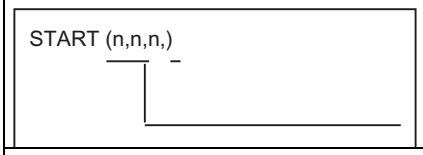
If several channels are involved in the machining of a workpiece it may be necessary to synchronize the programs.

There are special statements (commands) for this program coordination. In each case, they are listed in one block.

References:

/PG/ Programming Manual Job Planning; Flexible NC programming.

Table 6- 1 Program coordination statements

Statement	Description
	Selection of a program for processing in a certain channel: Acknowledgment mode: **n** (without) or **s** (synchronous) Name of the program with specification of the path Number of channel: Values 1 to 4 possible
CLEAR (identifier)	Deletion of a program by indicating the program identifier
	Starting the selected programs in other channels. Enumeration of the channel numbers: Values 1 to 4 possible
WAITM (Mnr, n, n, n, n)	Waiting for tag number Mnr for synchronization in the specified channels n (own channel can be, but must not be specified). The tag number must be the same in all channels. Numbers 0 to 9 are possible.
WAITE (n,n,n)	Waits for the end of program of the specified channels (current channel not specified)
SETM(Mnr1, Mnr2, ...Mnri)	Set wait marks Mnr1, Mnr2, ...Mnri for conditional wait with WAITMC() for the channel in which SETM() is issued. The channel thus declares its wait characteristics for the partner channels as fulfilled. The command can be activated in synchronized actions. Up to 10 marks (0-9) can be set using one command.
CLEARM(Mnr1, Mnr2, ...Mnri)	Delete wait marks Mnr1, Mnr2, ...Mnri for conditional wait with WAITMC() for the channel in which CLEARM() is issued. The channel thus declares to its partner channels that its wait characteristic is fulfilled. The command can be activated in synchronized actions. Up to 10 marks (0 - 9) can be deleted using one command.
WAITMC(Mnr, n1, n2, ...)	Conditional wait in continuous-path mode for the specified wait characteristic Mnr from the specified channels n1, n2, ... nk. The current channel can be specified, but this is optional. When processing continues after the wait marks from the other channels in the group have arrived, the wait marks of these channels are deleted.

The number of markers depends on the CPU used.

CPU 572 --> 2 Channels --> = 20

CPU 573 --> 10 Channels --> = 100

Behavior up to SW version 3

When a WAITM() call is reached, the axes in the current channel are decelerated and the system waits until the tag number specified in the call is received from the other channels to be synchronized. The group is synchronized when the other channels are also decelerated as they reach their WAITM() command. The synchronized channels then continue operation.

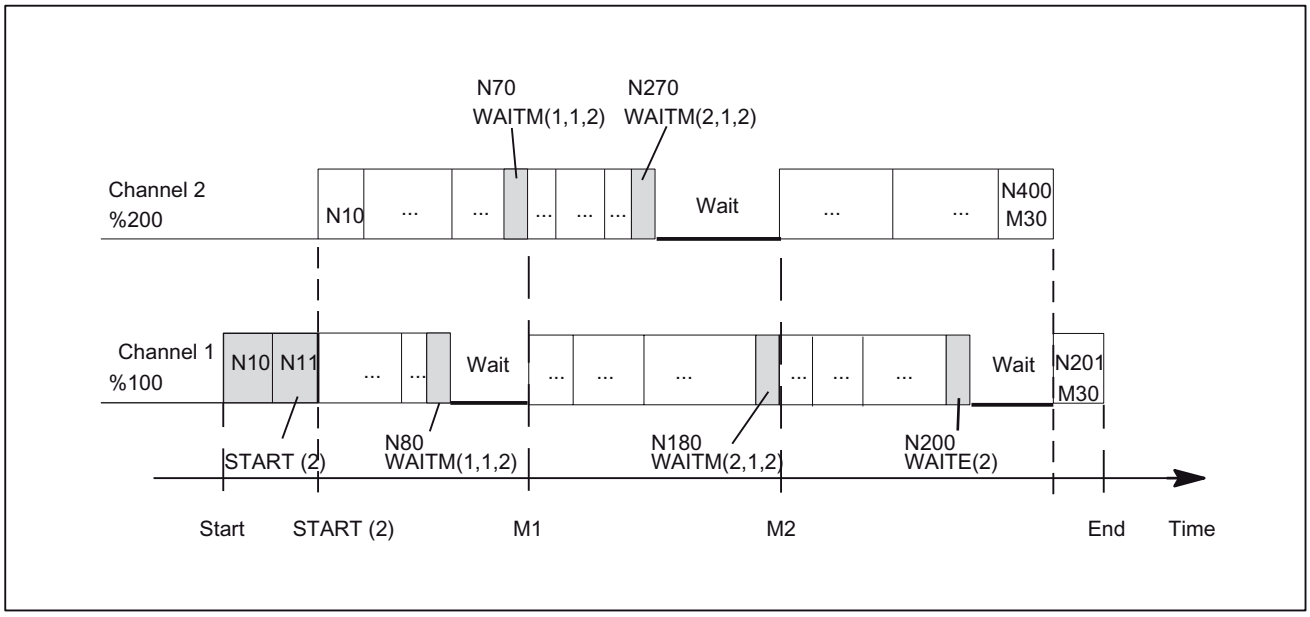


Figure 6-1 Program flow illustrated with example, coordination with WAITM(), unconditional wait

Example of program coordination

Channel 1:

```
%100
N10 INIT(2, "_N_200_MPF","n")
N11 START(2)
.                                   ; Processing in channel 1

N80 WAITM(1,1,2)                    ; Wait for WAIT-tag 1 in the channel 1 and
                                    in the channel 2

.                                   ; additional processing in Channel 1

N180 WAITM(2,1,2)                   ; Wait for WAIT-tag 1 in the channel 2 and
                                    in the channel 2

.                                   ; additional processing in Channel 1
N200 WAITE(2)                       ; Wait for end of program in channel 2
N201 M30                            ; Program end channel 1, total end
.
.
```

Channel 2:

```
%200
.                                   ; Processing in channel 2

N70 WAITM(1,1,2)                    ; Wait for WAIT-tag 1 in the channel 1 and
                                    in the channel 2

.                                   ; additional processing in Channel 2

N270 WAITM(2,1,2)                   ; Wait for WAIT-tag 1 in the channel 2 and
                                    in the channel 2

.                                   ; additional processing in Channel 2

N400 M30                            ; End of program in channel 2
.
.
```

6.3.2 Conditional wait in continuous path mode WAITMC

Objectives

Decelerating and waiting must take place only in cases where not all the channels to be coordinated have set their mark numbers for the purpose of synchronization. Conditional waiting.

The instants in time for generating wait marks and the conditional wait calls are decoupled.

For the purpose of inter-channel communication, marks may even be set when waiting and decelerating are not intended at all. No WAITMC() command. In this case, the channel marks settings remain valid after execution of RESET and NC Start.

Preconditions for conditional wait

To utilize conditional wait with WAITMC() and reduced wait times, the following conditions must be fulfilled:

- Continuous-path mode G64 must be set
- Look Ahead function must be active
- exact stop (G60, G09) **not** selected.

  If exact stop is selected, waiting with WAITMC() corresponds to waiting with WAITM() from SW 3.

Response

A) Starting with the motion block before the WAITMC() call, the wait marks of the other channels to be synchronized are checked. If these have all been supplied, then the channels continue to operate without deceleration in continuous-path mode. No wait. The path velocity remains unchanged.

B) If at least one wait mark from one of the channels to be synchronized is missing, then the axes start to decelerate from path velocity down to exact stop velocity. A check is now performed in every interpolation cycle to see whether the missing wait marks of the channels to be coordinated have arrived in the meantime. If this is the case, the axis is accelerated up to path velocity again and machining continued.

C) If the marks to be supplied by the channels to be synchronized have not arrived by the time exact stop velocity is reached, the machining operation is halted until the missing marks appear. When the last required mark appears, the axes are accelerated from standstill up to path velocity.

The following table shows the sequences of events for cases A) - C):

Deceleration response to conditional wait with WAITMC()		
With WAITMC	**Response**	**Velocity curve**
A) Wait marks of all channels have already arrived	continued operation with no deceleration	
B) All wait marks arrived during deceleration from path velocity down to exact stop velocity	Deceleration ceases immediately when last expected wait mark appears. The axes are accelerated back up to path velocity.	
C) The last wait mark does not arrive until exact stop velocity has been reached.	Brake down to exact stop velocity. When the last required mark appears, the axes are accelerated from exact stop velocity up to path velocity.	

Extended behavior and block change when WAITMC occurs

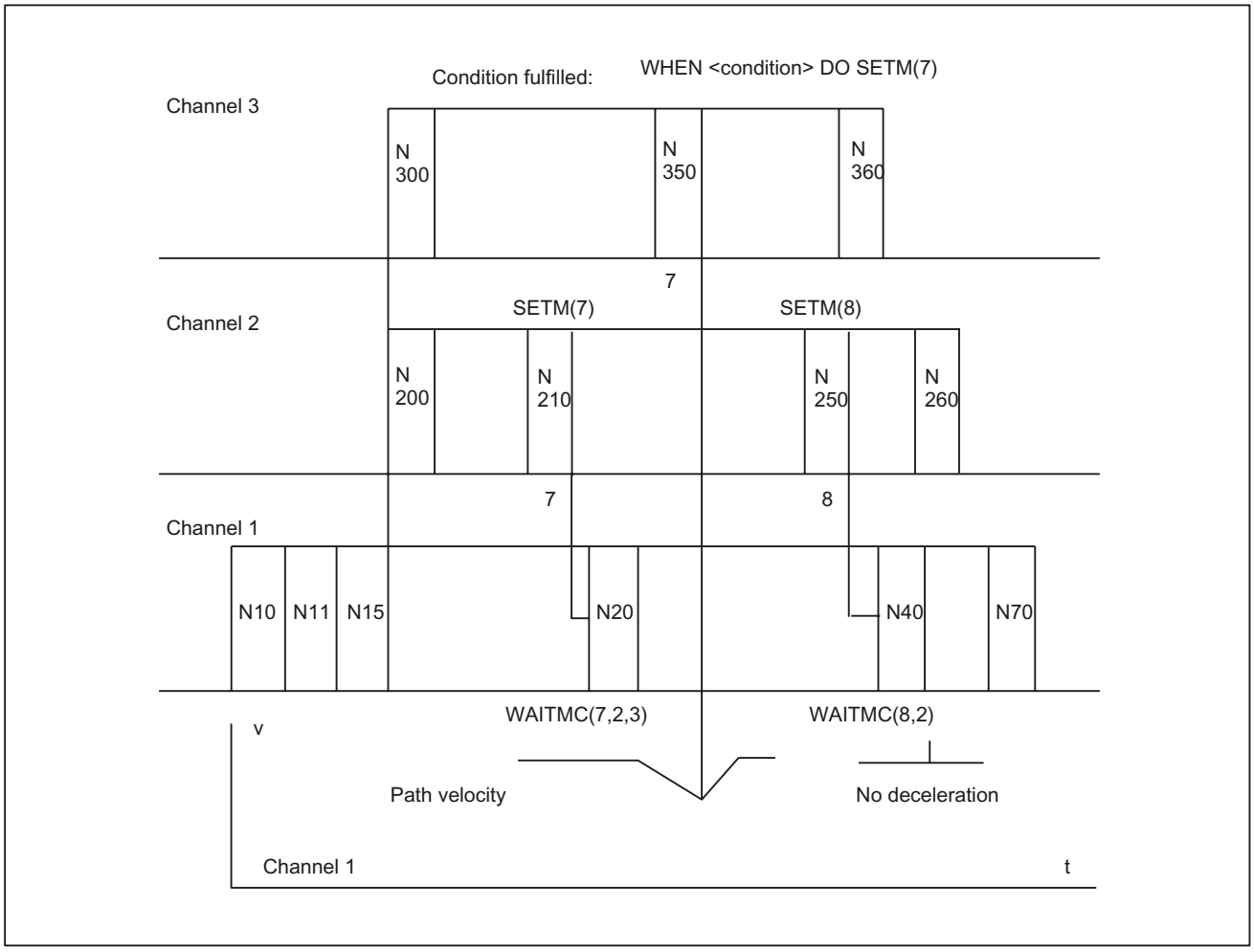
WAITMC and SETM (in Synact) can be synchronized.

Note

When G64 is active, a WAITMC(1,2,3) block does not generate **a separate block**, but is appended to the preceding block. A drop in velocity must be prevented when continuous-path mode is active. A WAITMC is therefore fulfilled if the preceding block is halted, e.g. by a read-in disable.

With block change condition IPOBRKA, when the wait flag is received, the next block is loaded instantaneously and the axes started, provided none of the other block end conditions prevent the block change. Braking only occurs if the flag is not yet reached, or another block end condition prevents the block change.

Example of conditional wait in continuous-path mode



Conditional wait involving three channels (schematic)

The example is schematic and shows only those commands that are relevant to the synchronization process.

Channel 1:

```
%100
N10 INIT(2, "_N_200_MPF","n")            ; select partner program Channel 2
N11 INIT(3, "_N_300_MPF","n")            ; select partner program Channel 3
N15 START(2, 3)                          ; start programs in Channel 2, 3
...                                      ; Processing in Channel 1:
N20 WAITMC(7, 2, 3)                      ; wait conditionally for marker 7 from
                                         Channels 2 and 3
...                                      ; Additional processing in Channel 1:
N40 WAITMC(8, 2)                         ; wait conditionally for marker 8 from
                                         Channel 2
...                                      ; Additional processing in Channel 1:
N70 M30                                  ; End Channel 1
```

Channel 2:

```
%200
N200                                    ; Processing in Channel 2
N210 SETM(7)                            ; Channel 2 sets wait marker 7
...                                     ; Additional processing in Channel 2
N250 SETM(8)                            ; Channel 2 sets wait marker 8
N260 M30                                ; End Channel 2
```

Channel 3:

```
%300
N300                                    ; Processing in Channel 3
...
N350 WHEN <condition> DO SETM(7)
                                        ; set wait marker in a synchronous action
                                        ; Additional processing in Channel 3
N360 M30                                ; End Channel 3
```

Example of WAITMC and read-in disabled

M555 is output in channel 3 while the axis is traversing and generates a read-in disabled (RID). As the WAITMC is appended to BLOCK N312, the wait mark is set and processing in channel 2 continues.

Channel 2:

```
N112 G18 G64 X200 Z200 F567             ; Processing in Channel 2
N120 WAITMC(1,2,3)                      ; Channel 2 sets wait markers 1, 2 and 3
...                                     ; Additional processing in Channel 2, as
...                                     ; WAITMC is added to block N312.
...                                     ; Additional processing in Channel 2
N170 M30                                ; End Channel 2
```

Channel 3:

```
                                        ; during travel read-in disabled M555
N300                                    ; Processing in Channel 3
N312 G18 G64 D1 X180 Z300 M555
N320 WAITMC(1,2,3)                      ; wait because of RID
```

Wait mark 1 is set in Channels 2 and 3

Channel 2 proceeds with additional processing and program processing in Channel 3 is stopped because of read-in disabled.

This behavior can be transferred to all available channels.

6.4 Axis/spindle replacement

6.4.1 Introduction

General information

An axis/a spindle is permanently assigned to a specific channel via machine data. The axis/spindle can be used in this channel only.

Definition

With the function "Axis or spindle replacement" it is possible to enable an axis or a spindle and to allocate it to another channel, that means to replace the axis/spindle.

Since the spindle function is subordinated to the axis function, only the term "Axis replacement" is used in the following.

Axis types

According to the channel, we distinguish four types of axes: The reactions at axis change depend on the settings in MD 30552:

MD30552 $MA_AUTO_GET_TYPE

Channel axis

A channel axis can be programmed in the parts program and traversed in all modes.

PLC axis

A PLC axis can only be positioned via the PLC.

If a PLC axis is programmed in the parts program

- in case of MD AUTO_GET_TYPE = 0 an alarm will be output.
- in case of MD AUTO_GET_TYPE = 1 an automatic GET will be generated.
- in case of MD AUTO_GET_TYPE = 2 an automatic GETD will be generated.

Neutral axis

If a neutral axis is programmed in the parts program

- in case of MD AUTO_GET_TYPE = 0 an alarm will be output.
- in case of MD AUTO_GET_TYPE = 1 an automatic GET will be generated.
- in case of MD AUTO_GET_TYPE = 2 an automatic GETD will be generated.

Axis in another channel

This is actually not a proper type of axis. It is the internal state of a replaceable axis. If this happens to be active in another channel (as channel, PLC or neutral axis).

If an axis is programmed in another channel in the parts program:

- in case of MD AUTO_GET_TYPE = 0 an alarm will be output.
- in case of MD AUTO_GET_TYPE = 1 an automatic GET will be generated.
- in case of MD AUTO_GET_TYPE = 2 an automatic GETD will be generated.

Note

Both machine data:

MD20110 $MC_RESET_MODE_MASK

MD20112 $MC_START_MODE_MASK

control the behavior of axis assignments in RESET, during booting and parts program start. The settings for channels between which axes are to be replaced must be selected such that no illegal constellations (alarms) are generated in conjunction with the following machine data:

MD30552 $MA_AUTO_GET_TYPE

References:

/FB1/ Function Manual Basic Functions; " ... Workpiece-related system of actual values" (K2)

Requirements

In order for an axis replacement to be executed, the valid channel for the machine axis number must be specified via the **channel-specific** machine data

MD20070 $MC_AXCONF_MACHAX_USED (machine axis number valid in channel)

and the delete setting of the channel for axis replacement via the **axis-specific** MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN (channel for axis replacement).

This results in the following regulations:

1. In which channel can the axis be used and replaced?
2. To which channel shall the axis be allocated with power ON?

Example of an axis replacement between channels

With 6 axes and 2 channels, the 1st, 2nd, 3rd and 4th axis in channel 1 and the 5th and 6th axis in channel 2 shall be used. It shall be possible to replace the 1st axis, this shall be allocated to channel 2 after power ON.

The **channel-specific** machine data must be allocated with:

CHANDATA(1)
MD20070 $MC_AXCONF_MACHAX_USED=(**1**, 2, 3, 4, 0, 0, 0, 0)

CHANDATA(2)
MD20070 $MC_AXCONF_MACHAX_USED=(5, 6, **1**, 0, 0, 0, 0, 0)

The **axis-specific** machine data must be allocated with:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[AX1]=2

Display

The current type of axis and the current channel for this axis will be displayed in an axial PLC interface byte. See Section "Axis replacement by PLC".

Note

If an axis is not valid in the channel selected, this is displayed by inversion of the axis name on the operator panel front of HMI.

6.4.2 Example of an axis replacement

Assumption

With 6 axes and 2 channels, the 1st, 2nd, 3rd and 4th axis in channel 1 and the 5th and 6th axis in channel 2 shall be used. It shall be possible to replace the 2nd axis between the channels and to allocate to channel 1 after power ON.

Exercise

The task is subdivided into the following areas:

- Machine data allocation so that the prerequisites for axis replacement are given.
- Programming of axis replacement between channel 1 and channel 2.

Fulfillment of preconditions

Population of channel-specific machine data MD20070

$MC_AXCONF_MACHAX_USED[1]=(1, **2**, 3, 4, 0, 0, 0, 0)

$MC_AXCONF_MACHAX_USED[2]=(5, 6, **2**, 0, 0, 0, 0, 0)

Population of the axis-specific machine data:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN[AX2]=1

Program in channel 1

```
...
RELEASE (AX2)
        ; Release the axis AX2
INIT (2, "_N_MPF_DIR\_N_TAUSH2_MPF", "S")
        ; Select program TAUSH2 in channel 2
START (2)
        ; Start program in channel 2
WAITM(1,1,2)
        ; Wait for wait flag 1 in channels 1 and 2
...     ;
...     ;Rest of program after axis replacement
...     ;
...     ;
M30
```

Program TAUSH2 in channel 2

```
...
WAITM(1,1,2)
        ; Wait for wait flag 1 in channels 1 and 2
GET (AX2)
...     ; Accept the axis AX2
...     ;
...     ;Rest of program after axis replacement
...     ;
RELEASE (AX2)
...     ; Release for additional axis replacement
...     ;
M30
```

6.4.3 Axis replacement options

One or more axes/spindles can be activated for replacement between channels by the parts program or by motion-synchronous actions. An axis/spindle replacement can also be requested and released from the PLC via the VDI interface. The axis/spindle must have been released in the current channel and will be taken over by the other channel with the GET command and released with the RELEASE command.

If the specified conditions are met, an axis/spindle replacement is initiated by:

- Programming in parts program GET/GETD

  Take over an axis or a spindle from another channel with GET or get it from another channel with GETD. With GETD a corresponding release is not required.
- Automatic by programming of the axis name, if the required conditions have been met by MD30552 $MA_AUTO_GET_TYPE > 0.
- Without preprocessing stop and existing synchronization between preprocessing and main run.
- Through PLC via the VDI interface to the NCK.

When taking over a PLC-controlled axis, the channel behavior triggered by the NC program can be de-coupled via an interface signal. This allows individual PLC axes to be interpolated independently from the NC program.

References:

/FB2/ Function Manual Extended Functions; Positioning Axes (P2)
Chapter "Independent Single Axis Procedures".

Axis replacement extensions

- Setting axis replacement behavior as variable via machine data MD10722 $MN_AXCHANGE_MASK.
- Axis replacement with an axis container rotation with implicit GET/GETD
- Axis replacement without pre-processing stop of axes not involved in the contour.
- Geometry axis with rotated frame and axis replacement in JOG mode via machine data MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED can be activated.
- Axis replacement via synchronized actions GET(axis), RELEASE(axis), AXTOCHAN, $AA_AXCHANGE_TYP(axis).

6.4.4 Replacement behavior NC program

Possible transitions

The following diagram shows which axis replacement possibilities are available.

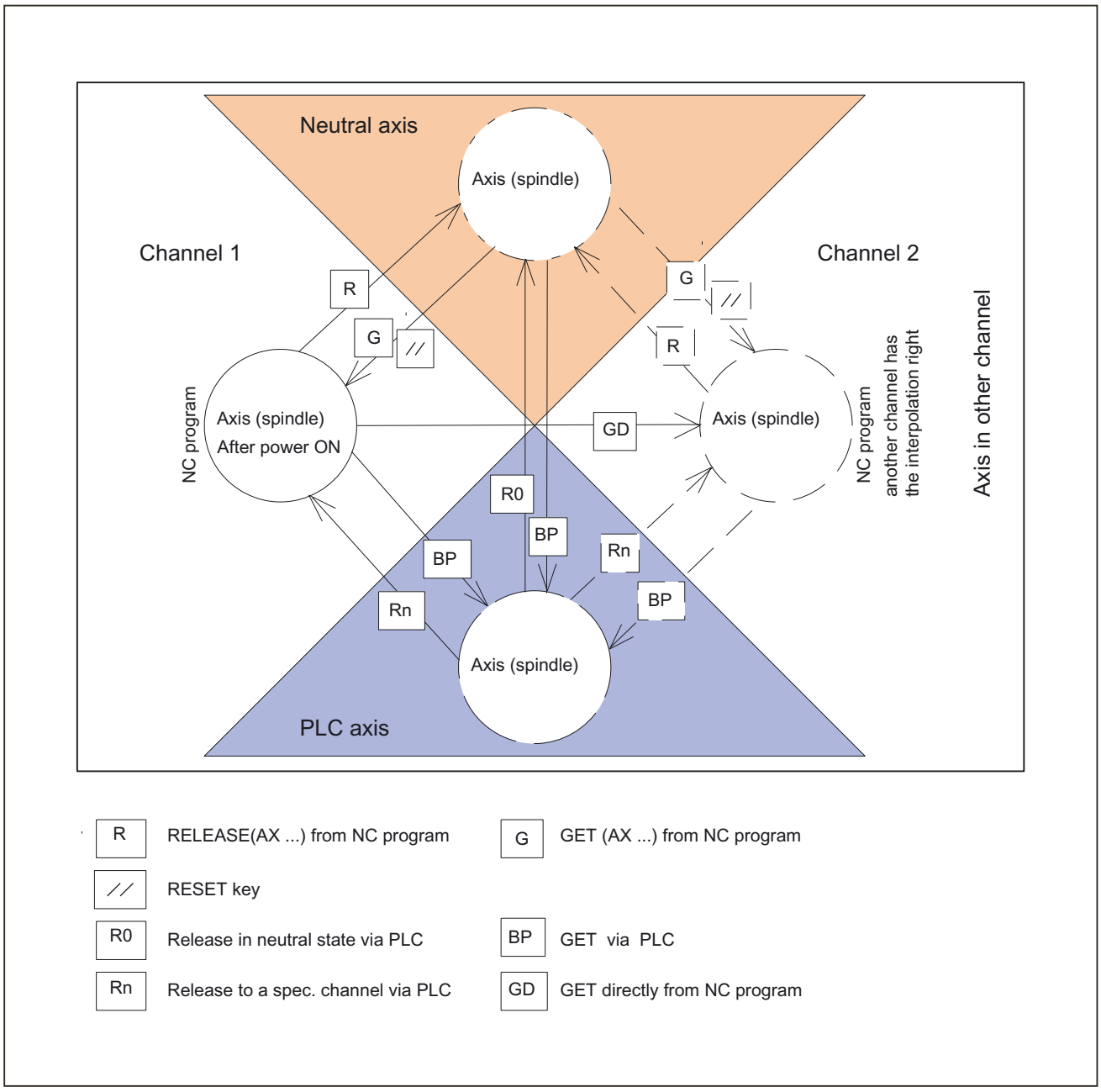


Figure 6-2 Transitions of possible axis states during axis replacement

6.4.5 Axis transfer to neutral state (release)

RELEASE

Notation in parts program:

RELEASE (axis name, axis name, SPI (Spindle no.), ...)

Note

The axis name corresponds to the axis allocations in the system and is either

- AX1, AX2, AX3, ... **or**
- the name assigned via the following machine data:
  MD10000 $MN_AXCONF_MACHAX_NAME_TAB

With RELEASE (axis name, ...) a dedicated NC block will always be generated.

Exception: The axis is already in the neutral state.

The RELEASE command is interrupted if

- the prerequisites for axis replacement are not fulfilled
  (MD20070 $MC_AXCONF_MACHAX_USED and
  MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN)
- the axis is involved in a transformation
- the axis is within an axis network.

Note

If the RELEASE command is applied to a gantry master axis, all following axes are released, too.

If ...	and ...	then ...
the axis is released, but not yet transferred with GET ...	... a RESET takes place via the operator panel front ...	... the axis is allocated again to the last responsible channel.

6.4.6 Transferring axis or spindle in the part program

Options

The release time and the behavior of an axis or spindle replacement is influenced in the part program as follows:

– Programming with the GET command in the same channel.

– Directly from another channel through programming with GETD.

References:
/PGA/ Programming Manual Job Planning; Chapter "Axis Replacement, Spindle Replacement (RELEASE, GET, GETD)"

With the GETD command

GET (axis name, axis name, SPI (spindle no.), ...)

The takeover of an axis is delayed if

- the axis is changing the measuring system.
- servo disable is being processed for the axis (transition from control in follow-up/stop and vice versa).
- the axis/spindle disable is set.
- the axis has not yet been enabled by the other channel with RELEASE.
- interpolation for the axis has not yet been completed (except for a speed-controlled spindle).

With GET (axis name, ...) a separate NC block with pre-processing stop is always generated.

Exceptions:

- If the axis is already a channel axis, then no block is generated.
- If the axis is synchronous, (i.e. it has not been swapped to another channel in the meantime or received a signal from the PLC) no extra block is generated either.

With the GETD command

With **GETD** (GET Directly) an axis is fetched directly from another channel. That means that no suitable RELEASE must be programmed for this GETD in another channel. In addition, another channel communication must be created (e.g. wait marks), since the supplying channel is interrupted with GETD1. If the axis is a PLC axis, replacement is delayed until the PLC has enabled the axis.

⚠ CAUTION
This programming command interrupts the program run in the channel in which the required axis is currently to be found! (REORG). Exception: The axis is at the time in a neutral state.

Note

If the GET or GETD command has been programmed, take-over is delayed and the channel is reset; the channel will no longer try to take over the axis.

An axis assumed with GET remains allocated to this channel even after a key RESET or program RESET. The axis can be replaced by programming RELEASE and GET again or will be assigned to the channel defined in the following machine data during POWER ON:

MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN

6.4.7 Automatic axis replacement

Automatically through programming of axis name

Depending on machine data MD30552 $MA_AUTO_GET_TYPE a GET or GETD is generated automatically if a neutral axis is again programmed or assigned to another channel.

Preconditions for automatic axis replacement

MD30552 $MA_AUTO_GET_TYPE > 0 must be true for automatic axis replacement.

Automatic GETD

Note

If an automatic GETD is set, the following must be observed:

- Channels could mutually influence each other.
  (REORG, when axis is removed.)
- With simultaneous access of several channels to an axis it is not known which channel will have the axis at the end.

Example 1

```
N1 M3 S1000
N2 RELEASE (SPI(1))                    ; => Transition to neutral condition
N3 S3000                               ; New speed for released spindle
                                       ; MD AUTO_GET_TYPE =
                                       ; 0 =>Alarm "Wrong axis type" is
                                       generated
                                       ; 1 => GET (SPI(1)) is generated.
                                       ; 2 => GETD (SPI(1)) is generated.
```

Example 2

```
                                       ; (axis 1 = X)
N1 RELEASE (AX1)                       ; => Transition to neutral condition
N2 G04 F2
N3 G0 X100 Y100:                       ; Motion of the released axis
                                       ; MD AUTO_GET_TYPE =
                                       ; 0 =>Alarm "Wrong axis type" is
                                       generated
                                       ; 1 => GET (AX1) is generated.
                                       ; 2 => GETD (AX1) is generated.
```

Example 3

```
                                       ; (axis 1 = X)
N1 RELEASE (AX1)                       ; => Transition to neutral condition
N2 G04 F2
N3 POS (X) = 100:                      ; Positioning of the released axis:
                                       ; MD AUTO_GET_TYPE =
                                       ; 0 =>Alarm "Wrong axis type" is
                                       generated
                                       ; 1 => GET (AX1) is generated. *)
                                       ; 2 => GET (AX1) is generated. *)
```

*) If the axis is still synchronized, no dedicated block will be generated.

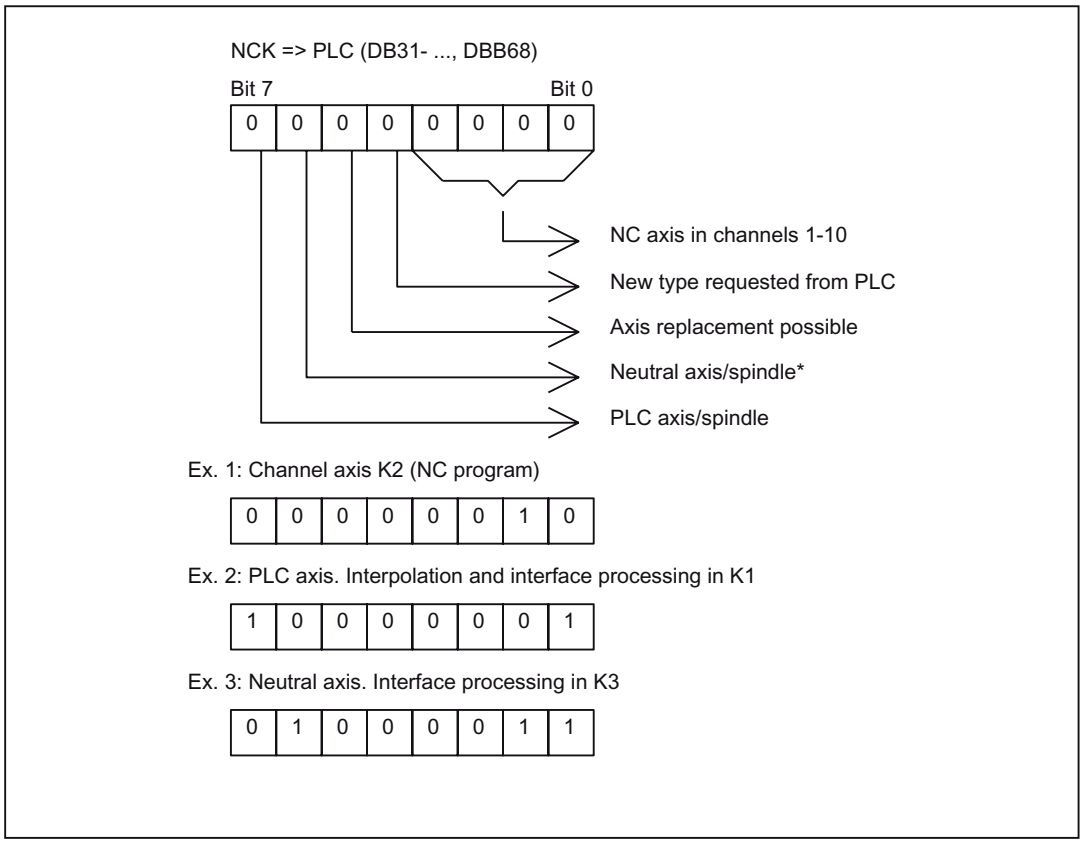
6.4.8 Axis replacement via PLC

The PLC can request and traverse an axis at any time and in any operating mode.

The PLC can change an axis from one channel into the other channel (only for 840D sl).

TYPE display

The type of an axis can be determined at any time via an interface byte (PLC axis, channel axis, neutral axis).



*** neutral axis/spindle** also contains the **command/oscillation axis**

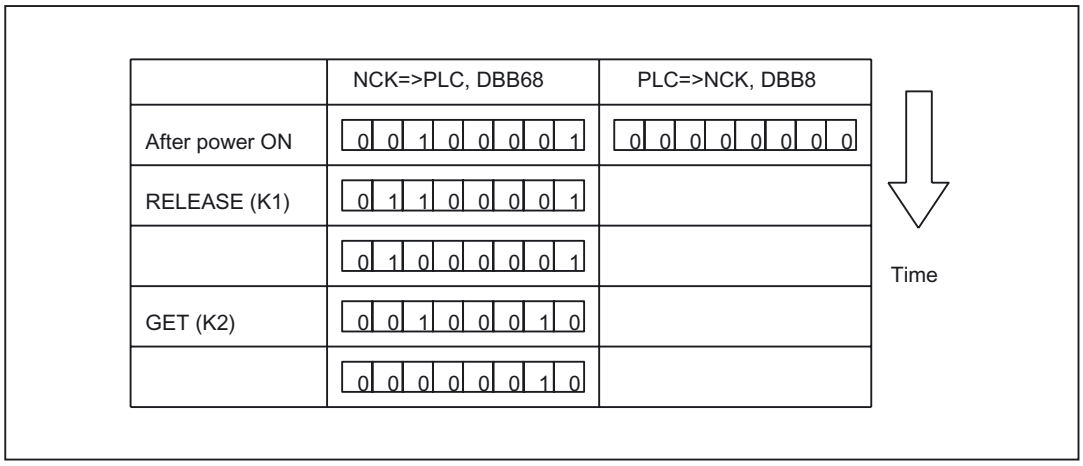
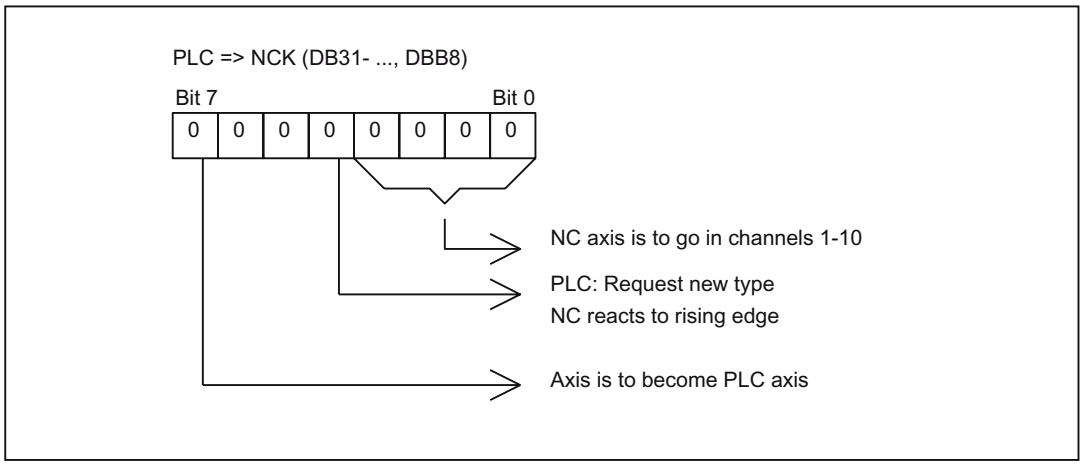


Figure 6-3 Changing an axis from K1 to K2 via parts program

TYPE input



In principle, the PLC must set the signal "Request new type". It is deleted again after change. Also for a channel interchange with GET and RELEASE (only 840D sl).

Controlling PLC axes/spindles for 840D sl

PLC axes and PLC spindles are traversed using function block FC18 in the basic PLC program

FC18: SpinCtrl Spindle control

Examples

The sequence of NC/PLC interface signals to change an NC axis to a PLC axis and to transition an NC axis into a neutral axis by the PLC are shown in the following diagrams.

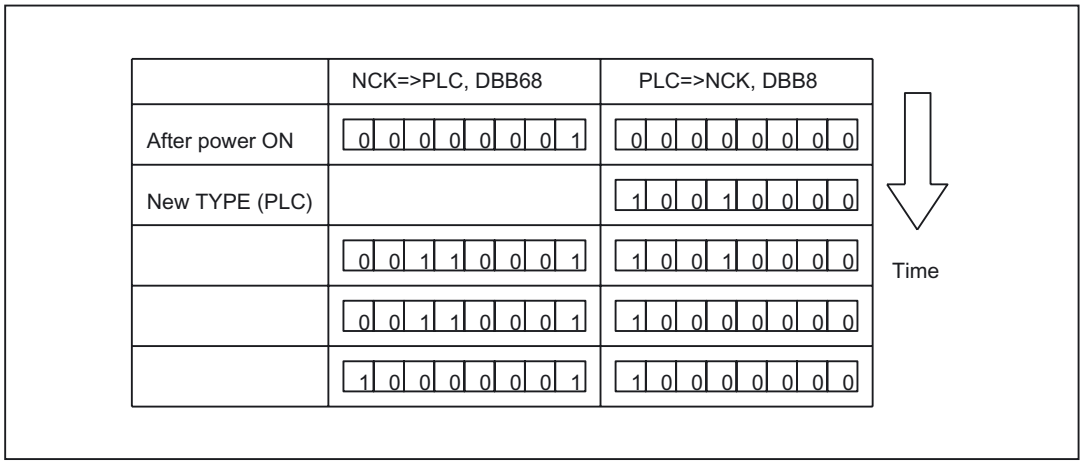


Figure 6-4 Changing an NC axis to a PLC axis

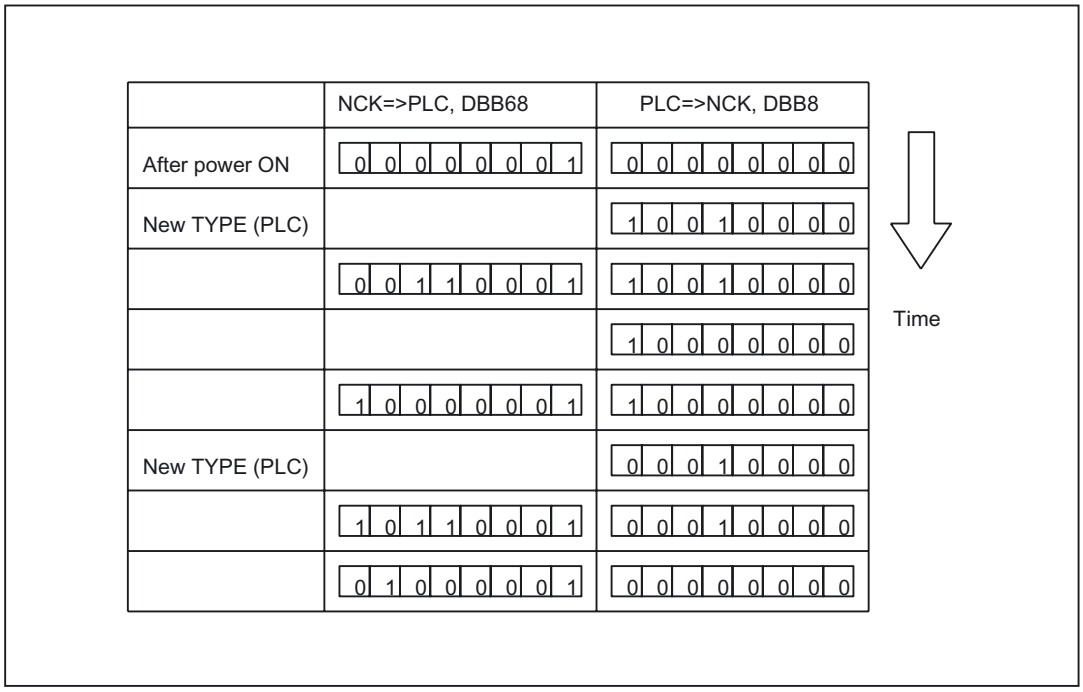


Figure 6-5 Changing an NC axis to a neutral axis through the PLC

6.4.9 Set axis replacement behavior variable.

The axis is replaced in the currently enabled channel and, depending on the respective axis type, the axis replacement behavior can be influenced via machine data MD10722 $MN_AXCHANGE_MASK:

Table 6- 2 Time of release of axes or spindles during replacement

MD10722	Axis replacement behavior
Bit 0 = 1	**Automatic Axis Replacement** between two channels also takes place when the axis has been brought to a neutral state by WAITP.
Bit 1 = 1	Release by **Axis Container Rotation** with implicitly generated GET/GETD When requesting an axis container rotation, all axes of the axis container that can be assigned to the executing channel are taken into the channel by means of an implicit GET or GETD. A subsequent replacement is only allowed after completion of the axis container rotation.
Bit 2 = 1	**Axis replacement without pre-processing** and possible forced re-organization of axes not involved in the contour. When an intermediate block is inserted in the main run, a check will be made to determine whether or not reorganization is required. Reorganization is only necessary if the axis states of this block do **not** match the current axis states.
Bit 3 = 0	PLC replacing of permanently **assigned PLC axis** An axis replacement can be requested from the PLC for each axis. The permanently assigned PLC axis only from neutral axis to PLC axis and vice versa.
Bit 3 = 1	Replacement request via VDI interface A replacement request via VDI interface is only executed for an: **axis exclusively controlled by the PLC** when MD30460 $MA_BASE_FUNCTION_MASK with Bit4=1. **permanently assigned** competing positioning axis (PLC axis) when MD30460 $MA_BASE_FUNCTION_MASK with Bit5=1. For such axes the interface signal NCK → PLC DB31, ... DBX68.5 (replacement possible) is always set to 1. For all other axes this bit is set to 0. For a permanently assigned PLC axis an axis replacement is only possible from neutral axis to PLC axis and vice versa.

6.4.10 Axis replacement via axis container rotation

Release axis container rotation

When an axis container rotation is released, all axis container axes that can be assigned to the channel are assigned to the channel by means of implicit GET or GETD commands. The axes can only be released again after the axis container rotation.

Note

This response does **not** apply if an axis in main run axis state (e.g. the PLC axis) is to take part in an axis container rotation, as this axis would have to give up this state for the axis container rotation.

References:

For more information about axis replacement of channel axes with axis containers, please see:

/FB2/ Function Manual, Extended Functions; Several Operator Panel Fronts and NCUs (B3)

Example: Axis container rotation with implicit GET or GETD

Action Channel 1	Action Channel 2
	SPOS = 180 positioned
axctswe(CT 1)	; gets spindle in Channel 1
	; and allows axis container rotation

Assumption:

The spindle is used in both channels and is also an axis in axis container CT 1.

Activation

Replacement via axis container rotation and implicit GET/GETD is activated with machine data MD10722 $MN_AXCHANGE_MASK, Bit 1=1.

6.4.11 Axis replacement with and without preprocessing stop

Axis replacement extension without preprocessing stop

Instead of a GET block with a preprocessing stop, this GET request only generates an intermediate block. In the main run, when this block is executed, the system checks whether the states of the axes in the block match the current axis states. If they do not match, forced reorganization can be triggered.

The following states of an **axis or positioned spindle** are checked for:

- The mode, either for the axis or for positioned spindle
- Setpoint position

The following states of a **Spindle in speed mode** are checked:

- Spindle mode: Speed mode
- Spindle speed S
- Direction of rotation M3, M4
- Gear stage M40, M41, M42, M43, M44, M45
- Master spindle at constant cutting rate.

In some instances, forced reorganization may be possible. Reorganization of the following axes is forced in any case.

Activation

Replacement without preprocessing and checking of the current states is activated with machine data MD10722 $MN_AXCHANGE_MASK, Bit 2=1.

Example

Activating an axis replacement without a preprocessing stop

```
N010 M4 S1000
N011 G4 F2
N020 M5
N021 SPOS=0
N022 POS[B]=1
N023 WAITP(B)                              ; Axis b becomes the neutral axis
N030 X1 F10
N031 X100 F500
N032 X200
N040 M3 S500
N041 G4 F2
N050 M5
N099 M30
```

If the spindle (axis B) is traversed immediately after block N023 as a PLC axis to 180° and back to 1°, and then again to the neutral axis, block N040 does not trigger a preprocessing stop nor a reorganization.

Special case: Axis replacement with preprocessing stop

Without a GET or GETD instruction having previously occurred in the main run, the spindle or the axis can be made available again by RELEASE (axis) or WAITP (axis), for example. A subsequent GET leads to a GET **with** a preprocessing stop.

6.4.12 Exclusively PLC-controlled axis and permanently-assigned PLC axis

Exclusively PLC-controlled axis

Activation

The exclusively PLC-controlled axis is activated via:

MD30460 $MA_BASE_FUNCTION_MASK, bit 4 = 1

Note

If an attempt is made to assign an exclusively PLC-controlled axis to the NC program or to request the axis for the NC program, then this is rejected with Alarm 26075. Analog to this, Alarm 26076 is signaled for a permanently-assigned PLC axis.

For an exclusively PLC-controlled axes, which is **not** a permanently assigned PLC axis, then the following functions are permitted:

- JOG motion of the axis via the traversing keys or handwheel
- Referencing the axis
- Traversing the axis via the function block **FC18** (for 840D sl)
- Traversing as permanently assigned PLC axis
- Traversing as oscillating axis (pre-condition: the "oscillation" option is available.)

  Reference:
  Function Manual Extended Functions; "Oscillation (P5)"

Axis control

If an axis is in the neutral state, it can be controlled by the signals AXRESET, AXSTOP and AXRESUME, effective in the channel.

Permanently assigned PLC axis

The permanently assigned PLC axis is activated via:

MD30460 $MA_BASE_FUNCTION_MASK, bit 5 = 1

During acceleration the axis becomes a neutral axis. A travel request via the NC/PLC interface converts a neutral axis without a previous axis interchange automatically to a concurrent positioning axis (PLC axis).

After traversing has been completed, if, for instance, the programmed end position has been reached, or if traversing has been interrupted, the axis again becomes the neutral axis. In this state, the axis can be moved or referenced using the traverse keys or the handwheel.

Note

A permanently assigned PLC axis cannot be replaced.

The following functions are **not** possible for a permanently assigned PLC axis:

- Replacement between channels
- Replacement to command axis
- Replacement to Oscillation axis
- Assignment of the axis to the NC program

The axis communicates its property as being a permanently-assigned PLC axis to the PLC via the following NC/PLC interface signal:

DB31, ... DBX71.7 (permanently-assigned PLC axis)

Axis control

If the axis is not PLC-controlled, it can be controlled by the signals RESET, STOP and RESUME, which are effective in the channel.

When booting, if it is identified that an axis defined as a geometry axis was simultaneously defined as permanently assigned PLC axis, then the axis function is signaled with Alarm 4320 "Axis %1 function MA_BASE_FUNCTION_MASK bit 5 and MC_AXCONF_GEOAX_ASSIGN_TAB not permitted".

If the change to a PLC axis is not possible

For a travel request from the NC/PLC interface area, a change to a PLC axis that is not possible is acknowledged at the NC/PLC interface using the start signal "Concurrent positioning axis active".

Note

Competing positioning axes get their block end points exclusively from the PLC an can be started at any point in time. Positioning axes are only influenced in their operation at block ends in the running parts program.

6.4.13 Geometry axis in rotated frame and axis replacement

Replacement expansion via Frame with Rotation

In JOG mode, a geometry axis with rotated frame can be traversed as PLC axis or as a command axis via static synchronized actions. In order to achieve this, in machine data MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED, bit 10=1 must be set. The reposition behavior of this axis is influenced via Bit 11.

Note

Before changing operational mode during JOG mode

Before changing the operational mode from JOG mode, all traverse motions of **all** PLC and command axes, which have been linked as geometry axes in the rotated frame, must have been concluded. These axes must at least have become neutral axes again, otherwise alarm 16908 will be generated when the operational mode is changed. This alarm is also generated when only a single geometry axis is traversed as a PLC or command axis in the rotated coordinate system.

Such an axis can only become a PLC or command axis within the channel, an axis replacement in another channel is not allowed.

Prerequisite for changing from JOG to AUTOMATIC

When changing from JOG mode to AUTOMATIC, the Condition program is interrupted and the end point of this geometry axis motion is only taken over if in MD 32074: FRAME_OR_CORRPOS_NOTALLOWED bit 11=1. This positions the PLC or command axes in relation to the rotation of the frame.

All axes influenced by a rotating frame are considered as geometry axes grouping and are handled collectively. In this way, all axes of the

- assigned to the NC program or
- all axes are neutral or
- are active as main run axes (PLC, command, or oscillation axis).

For example, if one axis is programmed with a WAITP, waiting is performed for all further axes of the geometry axis grouping, so that these axes can collectively become neutral axes. If one of the axes becomes a PLC axis in the main run, then all other axes of this grouping become neutral axes.

Boundary conditions

If in MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED bit 10=0 and in NC program **ROT Z45** is programmed, then **no axis replacement** is possible for the X axis and the Y axis. Similarly this applies to the Z axis, for instance with ROT X45 or ROT Y45 and also during JOG mode, when a block with such programming is interrupted. In this case, however, the signals for the X axis and the Y axis are at the VDI interface

IS DB31, ...DBX68.5 (”Replacement possible”) = 1
IS DB32, ...DBX68.5 (”Replacement possible”) = 1,

but both signals are reset to 0.

Only when in MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED bit 10=1 and no block currently interpolates with this programming, can a replacement for these axes be done in JOG mode.

6.4.14 Axis replacement from synchronized actions

Function

An axis can be requested with GET(axis) and be released for axis replacement with RELEASE(axis) with a synchronous action.

Note

The axis must be assigned as a channel axis via machine data.

An axis can be transferred directly between channels to a certain channel with the NC language command AXTOCHAN via synchronized actions or in the parts program. This axis does not have to be the same channel and it is not necessary that this channel be in possession of the current interpolation right for the axis.

Current state and interpolation right of the axis

With which axis type and interpolation right a possible axis replacement is to be performed, can be deducted from the system variable $AA_AXCHANGE_TYP[axis].

0: The axis is assigned to the NC program

1: Axis assigned to PLC or active as command axis or oscillating axis

2: Another channel has the interpolation right.

3: Axis is neutral axis.

4: Neutral axis is controlled by the PLC.

5: Another channel has the interpolation right, axis is requested for NC program.

6: Another channel has the interpolation right, axis is requested as neutral axis.

7: Axis active for PLC or as command or oscillating axis, axis is requested for PLC program.

8: Axis active for PLC or as command or oscillating axis, axis is requested as neutral axis.

9: Permanently assigned PLC axis, in state of neutral axis.

10: Permanently assigned PLC axis, controlled by PLC, in state of neutral axis.

Permanently assigned PLC axis

in state of neutral axis $AA_AXCHANGE_TYP = 9 and
controlled by PLC, in state of neutral axis $AA_AXCHANGE_TYP = 10

will be assigned to PLC **independently of GET and RELEASE** permanently.

Whether the axis can also be replaced is displayed via the system variable $AA_AXCHANGE_STAT[axis].

State transitions GET, RELEASE from synchronous actions and when GET is completed

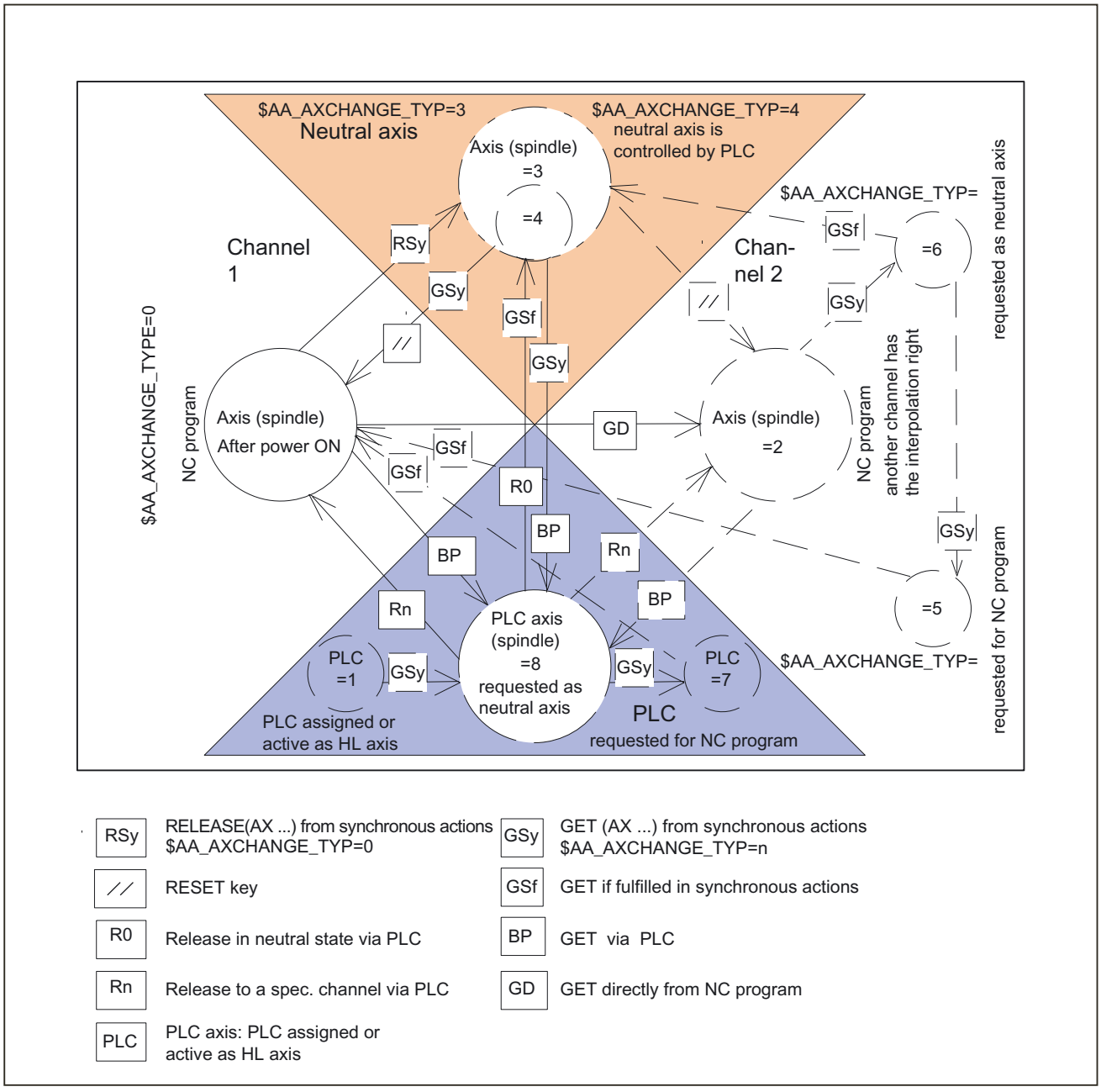


Figure 6-6 Transitions from synchronized actions

For more information, please refer to:

References:

/FBSY/ Function Manual, Synchronized Actions; "Actions in Synchronized Actions"
/PGA/ Programming Manual, Job Planning; "Motion-synchronous Actions"

6.5 Marginal conditions

Mode group

Up to 10 mode groups are available for SINUMERIK 840D sl.

Only 1 mode group is available for SINUMERIK 828D.

Channels

Up to 10 channels are available for SINUMERIK 840D sl.

Only 1 channel is available for SINUMERIK 828D.

Axis/spindle interchange

For SINUMERIK 828D, an axis/spindle interchange is not possible between channels.

Change to the channel axis

If an axis is changed from PLC axis, neutral axis or axis in another channel to the axis type channel axis, a synchronization must take place.

With this synchronization,

- the current positions are assumed
- the current speed and gear stage is assumed with spindles.

It is therefore obligatory to perform a feed stop which interrupts the active path movement.

If the axis is transferred with GET, this transition is clearly defined by the parts program.

If the axis is allocated by the PLC, the program section in which the change takes place is not clearly foreseeable.

(Except by a separate user-specific NC <-> PLC logic)

For this reason, the change to the channel axis is delayed in the following conditions:

- Path operation is active (G64+ axes programmed)
- Thread cutting/tapping is active (G33/G331/G332)

Change from a channel axis

The change of a channel axis to a neutral axis or PLC axes cannot be performed during an active path operation.

With RELEASE this is caused by the fact that RELEASE must be located in a separate NC block.

If the PLC changes the axis type, a REORG is triggered internally. Therefore, the change with the listed program conditions is delayed.

Block search

During block search with calculation, all GET, GETD or RELEASE blocks are stored and output after the next NC Start.

Exception:

Blocks which are mutually exclusive are deleted.

Example:

N10	RELEASE (AX1)	**Blocks are deleted.**
N40	GET (AX1)	"
N70	Target	

6.6 Data lists

6.6.1 Machine data

6.6.1.1 General machine data

Number	Identifier: $MN_	Description
10010	ASSIGN_CHAN_TO_MODE_GROUP[n]	Channel valid in mode group [Channel No.]: 0, 1
10722	AXCHANGE_MASK	Parameterization of the axis replacement response

6.6.1.2 Channel-specific machine data

Basic machine data of channel

Number	Identifier: $MC_	Description
20000	CHAN_NAME	Channel name
20050	AXCONF_GEOAX_ASSIGN_TAB[n]	Assignment of geometry axis to channel axis [GEOAxisNo.]: 0...2
20060	AXCONF_GEOAX_NAME_TAB[n]	Geometry axis name in channel [GEOAxisNo.]: 0...2
20070	AXCONF_MACHAX_USED[n]	Machine axis number valid in channel [Channel axis No.]: 0...7
20080	AXCONF_CHANAX_NAME_TAB[n]	Name of channel axis in the channel [Channel axis No.]: 0...7
20090	SPIND_DEF_MASTER_SPIND	Initial setting of master spindle in channel
20100	DIAMETER_AX_DEF	Geometry axis with transverse axis function
20110	RESET_MODE_MASK	Determination of basic control settings after Reset/TP End
20112	START_MODE_MASK	Determination of basic control settings after NC start
20150	GCODE_RESET_VALUES[n]	Reset G groups [G-Group No.]: 0...59
20160	CUBIC_SPLINE_BLOCKS	Number of blocks for C spline
20170	COMPRESS_BLOCK_PATH_LIMIT	Maximum traversing length of NC block for compression
20200	CHFRND_MAXNUM_DUMMY_BLOCKS	Empty blocks with phase/radii
20210	CUTCOM_CORNER_LIMIT	Max. angle for intersection calculation with tool radius compensation
20220	CUTCOM_MAX_DISC	Maximum value with DISC
20230	CUTCOM_CURVE_INSERT_LIMIT	Maximum angle for intersection calculation with tool radius compensation

Number	Identifier: $MC_	Description
20240	CUTCOM_MAXNUM_CHECK_BLOCKS	Blocks for predictive contour calculation with tool radius compensation
20250	CUTCOM_MAXNUM_DUMMY_BLOCKS	Max. no. of dummy blocks with no traversing movements with TRC
20270	CUTTING_EDGE_DEFAULT	Basic setting of tool cutting edge without programming
20400	LOOKAH_USE_VELO_NEXT_BLOCK	Look Ahead to programmed following block velocity
20430	LOOKAH_NUM_OVR_POINTS	Number of override switch points for Look Ahead
20440	LOOKAH_OVR_POINTS[n]	Override switch points for LookAhead [Switch point No.]: 0...1
20500	CONST_VELO_MIN_TIME	Minimum time with constant velocity
20600	MAX_PATH_JERK	Path-related maximum jerk
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements
20650	THREAD_START_IS_HARD	Acceleration behavior of axis with thread cutting
20700	REFP_NC_START_LOCK	NC start disable without reference point
20750	ALLOW_GO_IN_G96	G0 logic in G96
20800	SPF_END_TO_VDI	Subprogram end to PLC
21000	CIRCLE_ERROR_CONST	Circle end point monitoring constant
21010	CIRCLE_ERROR_FACTOR	Circle end point monitoring factor
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21200	LIFTFAST_DIST	Traversing path for fast retraction from the contour
21250	START_INDEX_R_PARAM	Number of first channel-specific R parameter

Auxiliary function settings of the channel

Number	Identifier: $MC_	Description
22000	AUXFU_ASSIGN_GROUP[n]	Auxiliary function group [aux. func. no. in channel]: 0...49
22010	AUXFU_ASSIGN_TYPE[n]	Auxiliary function type [aux. func. no. in channel]: 0...49
22020	AUXFU_ASSIGN_EXTENSION[n]	Auxiliary function extension [aux. func. no. in channel]: 0...49
22030	AUXFU_ASSIGN_VALUE[n]	Auxiliary function value [aux. func. no. in channel]: 0...49
22200	AUXFU_M_SYNC_TYPE	Output timing of M functions
22210	AUXFU_S_SYNC_TYPE	Output timing of S functions
22220	AUXFU_T_SYNC_TYPE	Output timing of T functions
22230	AUXFU_H_SYNC_TYPE	Output timing of H functions
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions
22250	AUXFU_D_SYNC_TYPE	Output timing of D functions

Number	Identifier: $MC_	Description
22260	AUXFU_E_SYNC_TYPE (available soon)	Output timing of E functions
22400	S_VALUES_ACTIVE_AFTER_RESET	S function active after RESET
22410	F_VALUES_ACTIVE_AFTER_RESET	F function active after reset
22500	GCODE_OUTPUT_TO_PLC	G functions to PLC
22550	TOOL_CHANGE_MODE	New tool offset for M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

Channel-specific memory settings

Number	Identifier: $MC_	Description
25000	REORG_LOG_LIMIT	Percentage of IPO buffer for log file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG (DRAM)
28010	MM_NUM_REORG_LUD_MODULES	Number of blocks for local user variables for REORG (DRAM)
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables (DRAM)
28030	MM_NUM_LUD_NAMES_PER_PROG	Number of local user variables per program (DRAM)
28040	MM_LUD_VALUES_MEM	Memory size for local user variables (DRAM)
28050	MM_NUM_R_PARAM	Number of channel-specific R parameters (SRAM)
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in IPO buffer (DRAM)
28070	MM_NUM_BLOCKS_IN_PREP	Number of blocks for block preparation (DRAM)
28080	MM_NUM_USER_FRAMES	Number of settable Frames (SRAM)
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles (DRAM)
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for compile cycles (DRAM)
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparation task (DRAM)
28510	MM_IPO_TASK_STACK_SIZE	Stack size of IPO task (DRAM)

6.6.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30460	BASE_FUNCTION_MASK	Axis functions
30550	AXCONF_ASSIGN_MASTER_CHAN	Reset position of channel for axis change
30552	AUTO_GET_TYPE	Definition of automatic GET
30600	FIX_POINT_POS	Fixed value positions of axes with G75
32074	FRAME_OR_CORRPOS_NOTALLOWED	Frame or HL offset are not allowed
33100	COMPRESS_POS_TOL	Maximum deviation with compensation

6.6.2 Setting data

6.6.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42000	THREAD_START_ANGLE	Start angle for thread
42100	DRY_RUN_FEED	Dry run feedrate

6.6.3 Signals

6.6.3.1 Signals to/from mode group

The mode group signals from the PLC to the NCK and from the NCK to the PLC are included in data block 11.

The signals are described in:
Reference:
Function Manual, Basic Functions; NC/PLC Interface Signals (Z1),
Chapter "Mode group, Program Operation (K1)"

6.6.3.2 Signals to/from channel

The channel signals from the PLC to the NCK and from the NCK to the PLC are included in data blocks 21, 22, ... for the first, second ... channel.

The signals are described in:
Reference:
Function Manual, Basic Functions; NC/PLC Interface Signals (Z1),
Chapter "Mode group, Program Operation (K1)"

M1: Kinematic transformation

7

7.1 Brief description

7.1.1 TRANSMIT (option)

Note

The "TRANSMIT and peripheral surface transformation" option that is under license is required for the "TRANSMIT" function.

The "TRANSMIT" function permits the following services:

- Face-end machining on turned parts in the turning clamp
  - Holes
  - Contours
- A cartesian coordinate system can be used to program these machining operations.
- The control maps the programmed traversing movements of the Cartesian coordinate system onto the traversing movements of the real machine axes (standard situation):
  - Rotary axis (1)
  - Infeed axis vertical to the axis of rotation (2)
  - Longitudinal axis parallel to the axis of rotation (3)

    Linear axes (2) and (3) are perpendicular to one another.
- A tool center offset relative to the turning center is permitted.
- The velocity control makes allowance for the limits defined for rotary motion.
- A path in the cartesian coordinate system must not pass through the turning center point (this restriction applies to SW 2 and 3).

Other system variables

- The tool center point path can pass through the turning center point of the rotary axis.
- The rotary axis does not need to be a modulo axis.

7.1.2 TRACYL (option)

Note

The "TRANSMIT and peripheral surface transformation" option that is under license is required for the function "Cylinder surface transformation (TRACYL)".

The function "Cylinder surface transformation (TRACYL)" permits the following services:

Machining of

- Longitudinal grooves on cylindrical bodies,
- Transverse grooves on cylindrical bodies
- Arbitrary groove patterns on cylindrical objects.

The path of the grooves is programmed with reference to the unwrapped, level surface of the cylinder.

For machining, lathes with

- X-C-Z kinematics and
- X-Y-Z-C kinematics

are supported..

- The control transforms the programmed traversing movements of the cylinder coordinate system into the traversing movements of the real machine axes (standard applications X-C-Z kinematics TRAFO_TYPE_n = 512):
  - Rotary axis (1)
  - Infeed axis vertical to the axis of rotation (2)
  - Longitudinal axis parallel to the axis of rotation (3)

  Note

  Linear axes (2) and (3) are perpendicular to one another. The infeed axis (2) intersects the rotary axis. This constellation does not permit groove side offset.

- For groove side offset, X-Y-Z-C kinematics is required with the following axes (TRAFO_TYPE_n = 513):
  - Rotary axis (1)
  - Infeed axis vertical to the axis of rotation (2)
  - Longitudinal axis parallel to the axis of rotation (3)
  - Longitudinal axis (4) to supplement (2) and (3) to obtain a right-hand cartesian coordinate system.

  Note

  Linear axes (2), (3) and (4) are perpendicular to one another. This constellation permits groove wall corrections.

- The velocity control makes allowance for the limits defined for rotary motion.

`TRACYL`transformation, without groove side compensation, with additional longitudinal axis (cylinder surface curve transformation without groove side offset TRAFO_TYPE_n= 514)

- Transformation without groove side offset requires only a rotary axis and a linear axis positioned perpendicular to the rotary axis.
- If a machine provides an additional linear axis positioned perpendicular to the rotary axis and first linear axis, this can be utilized to improve tool offset.

7.1.3 TRAANG (option)

Note

The "Inclined axis" option under license is required for the function "Inclined axis (TRAANG)".

The function "Inclined axis (TRAANG)" is intended for grinding applications. It allows the following:

- Machining with inclined infeed axis.
- A cartesian coordinate system can be used for programming purposes.
- The control maps the programmed traversing movements of the Cartesian coordinate system onto the traversing movements of the real machine axes (standard situation): Inclined infeed axis.

7.1.4 Chained transformations

Introduction

Two transformations can be chained so that the motion components for the axes from the first transformation are used as input data for the chained second transformation. The motion parts from the second transformation act on the machine axes.

Chaining options

- The chain may include **two** transformations.
- The **second** transformation must be **"Inclined axis"** (TRAANG).
- The first transformation can be:
  - Orientation transformations (TRAORI), incl. universal milling head
  - TRANSMIT
  - TRACYL
  - TRAANG

For more information about the other transformations, please refer to:
References:
/FB3/ Function Manual, Special Function; 3- to 5-axis Transformations (F2).

7.1.5 Activating transformation machine data via parts program/softkey

Most of the machine data relevant to kinematic transformations were activated by POWER ON up to now.

Transformation machine data can also be activated via the parts program/softkey and it is not necessary to boot the control.

7.2 TRANSMIT (option)

Note

The TRANSMIT transformation described below requires that unique names are assigned to machine axes, channel and geometry axes when the transformation is active.

See

MD10000 $MN_AXCONF_MACHAX_NAME_TAB,

MD20080 $MC_AXCONF_CHANAX_NAME_TAB,

MD20060 $MC_AXCONF_GEOAX_NAME_TAB.

Besides this, no unequivocal assignments exist.

Task specification

Complete machining, see diagram:

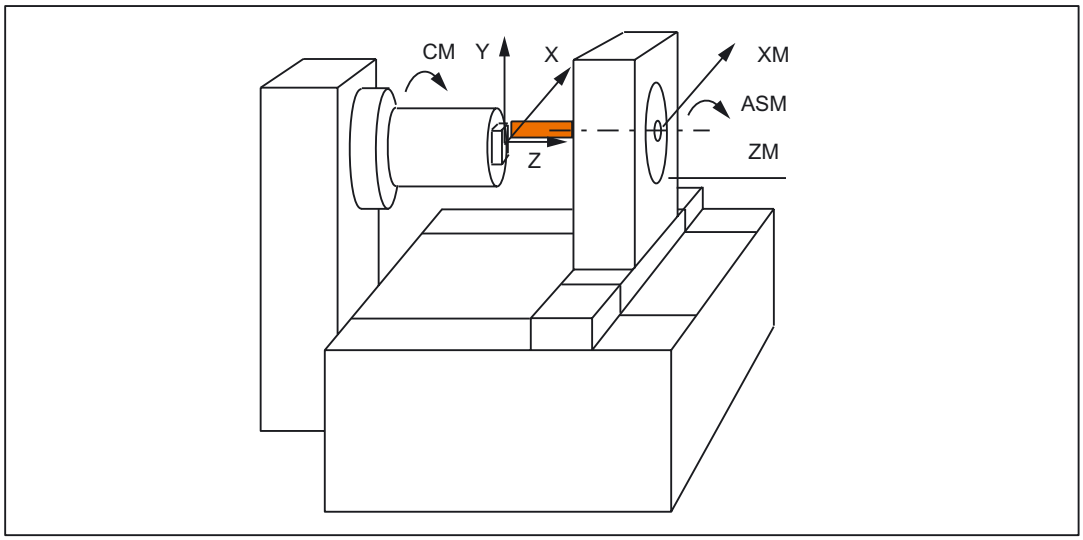


Figure 7-1 End face machining of turned part

Legend:

CM: Rotary axis (Main spindle)

ASM: Work spindle (miller, drill)

X, Y, Z: Cartesian coordinate system for programming of face-end machining (origin at the rotation center of the face)

ZM: Machine axis (linear)

XM: Machine axis (linear)

7.2.1 Preconditions for TRANSMIT

Axis configuration

Before movements can be programmed in the Cartesian coordinate system (according to Fig. X, Y, Z), the control system must be notified of the relationship between this coordinate system and the real machine axes (CM, XM, ZM, ASM):

- Assignment of names to geometry axes
- Assignment of geometry axes to channel axes
  - general case (`TRANSMIT` not active)
  - `TRANSMIT` active
- Assignment of channel axes to machine axis numbers
- Identification of spindles
- Allocation of machine axis names

With the exception of the "`TRANSMIT` active" point, the procedure is the same as for the normal axis configuration. If you already know the general steps, you need only read step "Assignment of geometry axes to channel axes" from the list of steps below.

References:

/FB1/ Function Manual Basic Functions; "Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset" (K2)

Number of transformations

Up to ten transformation data blocks can be defined for each channel in the system. Machine data names of these transformations begin with "$MC_TRAFO …" and end with "... _n", where n stand for a number from 1 to 10. The following sections include descriptions of these data:

MD24100 $MC_TRAFO_TYPE_n

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n

MD24110 $MC_TRAFO_AXES_IN_n.

Number of TRANSMIT structures

Two of the 10 permitted data structures for transformations in the channel may be assigned to the `TRANSMIT` function. They are characterized by the fact that the value assigned with the following machine data is 256 or 257:

MD24100 $MC_TRAFO_TYPE_n

The following machine data must be set for a maximum of 2 of these TRANSMIT transformations:

MD24950 $MC_TRANSMIT_ROT_AX_OFFSET_t

MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_t

MD24920 $MC_TRANSMIT_BASE_TOOL_t

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_t

In this case, t specifies the number of the declared TRANSMIT transformation (maximum of 2).

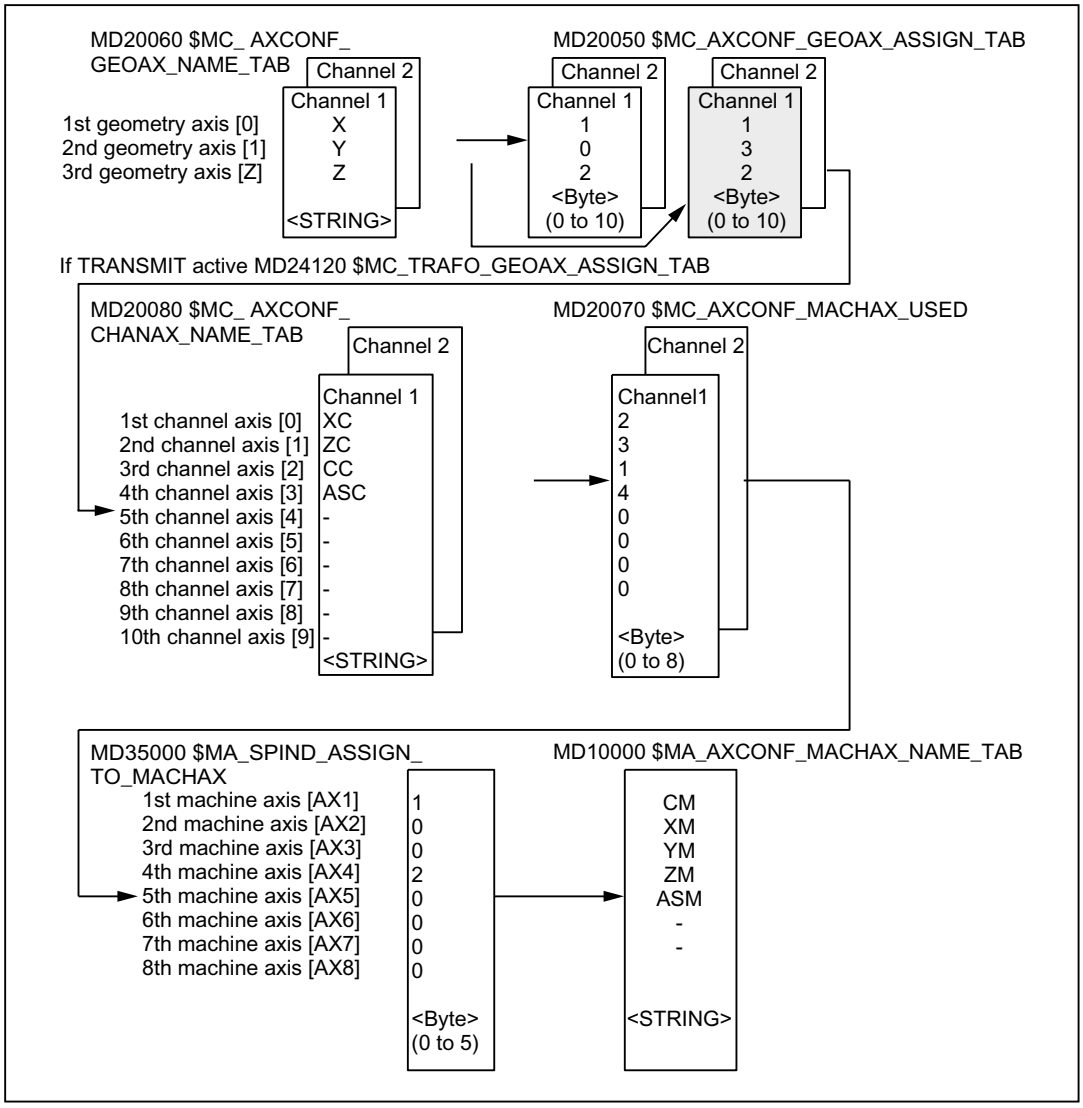


Figure 7-2 Axis configuration for the example in the figure "Face-end machining of turned part" (TRANSMIT)

The configurations highlighted in the figure above apply when TRANSMIT is active.

Assignment of names to geometry axes

According to the axis configuration overview shown above, the geometry axes involved in the TRANSMIT operation must be defined with:

MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"

MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"

MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"

(The choice of names in the above figure is in accordance with defaults).

Assignment of geometry axes to channel axes

A distinction has to be made, whether`TRANSMIT` is active or not:

- TRANSMIT not active

  One y-axis is not available.

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0]=1

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB_TAB[1]=0

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB_TAB[2]=2

- TRANSMIT active

  The Y-axis can be addressed by the parts program.

  MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1

  MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=3

  MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=2

  The Y-axis becomes the third entry of the channel axes.

Entry of channel axes

Those axes, which do not belong to the Cartesian coordinate system, are entered.

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="ZC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="CC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="ASC"

Assignment of channel axes to machine axes

With the cd of the channel axes as a reference, the machine axis number to which the channel axes have been assigned, is transferred to the control system.

MD20070 $MC_AXCONF_MACHAX_USED[0]=2

MD20070 $MC_AXCONF_MACHAX_USED[1]=3

MD20070 $MC_AXCONF_MACHAX_USED[2]=1

MD20070 $MC_AXCONF_MACHAX_USED[3]=4

MD20070 $MC_AXCONF_MACHAX_USED[4]=0

(entries corresponding to the figure above)

Identification of spindles

It is specified per machine axis, whether a spindle is present (value > 0: spindle number) or a path axis (value 0).

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0]=1

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1]=0

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2]=0

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3]=2

Assignment of names to machine axes

With the cd of the machine axes as a reference, a machine axis name is transferred to the control system.

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="ZM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ASM"

7.2.2 Settings specific to TRANSMIT

Type of transformation

The following paragraph describes how the transformation type is specified.

TRAFO_TYPE_n

The user must specify the transformation type for the transformation data blocks (maximum n = 10). The value 256 must be set for `TRANSMIT` or the VALUE 257 for a rotary axis with supplementary linear axis.

Example for VALUE 256: MD24100 $MC_TRAFO_TYPE_1=256

The setting must be made before `TRANSMIT` or `TRANSMIT`(t) is called, where "t" is the number of the declared `TRANSMIT` transformation.

The `TRANSMIT` transformation requires only a rotary axis and a linear axis positioned perpendicular to the rotary axis. A real Y axis is used with transformation type 257 in order to compensate for a tool offset, for example.

Transformation type 257

Polar transformation with a rotary axis TRAFO_TYPE_n = 25710.04

Transformation with supplementary linear axis

If the machine has another linear axis which is perpendicular to both the rotary axis and the first linear axis, transformation type 257 can be used to apply tool offsets with the real Y axis. It is assumed that the working area of the second linear axis is small and is not to be used for the retraction of the part program.

The existing settings for MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n apply.

Axis image

The following paragraph describes how the transformation axis image is specified.

TRAFO_AXES_IN_n

Three channel axis numbers must be specified for the transformation data block n:

MD24110 $MC_TRAFO_AXES_IN_1[0]=channel axis number of the axis perpendicular to the rotary axis.

MD24110 $MC_TRAFO_AXES_IN_1[1]=channel axis number of the rotary axis.

MD24110 $MC_TRAFO_AXES_IN_1[2]=channel axis number of the axis parallel to the rotary axis.

Example for the configuration according to the figure "Face-end machining of turned part" (TRANSMIT):

MD24110 $MC_TRAFO_AXES_IN_1[0]=1

MD24110 $MC_TRAFO_AXES_IN_1[1]=3

MD24110 $MC_TRAFO_AXES_IN_1[2]=2

The setting must be made before TRANSMIT or TRANSMIT(t) is activated. Axis numbers must correspond with the channel axis sequence in

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n.

For transformation type 257 the following indices apply to

MD24110 $MC_TRAFO_AXES_IN_n[ ].

Meaning of indices in relation to base coordinate system (BCS):

- [0]: Cartesian axis perpendicular to rotary axis (in machine zero position, this axis is parallel to the linear axis which is positioned perpendicular to the rotary axis)
- [1]: Cartesian axis perpendicular to rotary axis
- [2]: Cartesian axis parallel to rotary axis (if configured)
- [3]: Linear axis parallel to index [2] in initial position of machine

Meaning of indices in relation to machine coordinate system (MCS):

- [0]: Linear axis perpendicular to rotary axis
- [1]: Rotary axis
- [2]: Linear axis parallel to rotary axis (if configured)
- [3]: Linear axis perpendicular to the axes of indices [0] and [1]

Rotational position

The rotational position of the Cartesian coordinate system is specified by machine data as described in the following paragraph.

TRANSMIT_ROT_AX_OFFSET_t

The rotational position of the x-y plane of the Cartesian coordinate system in relation to the defined zero position of the rotary axis is specified with:

MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_t= ... °

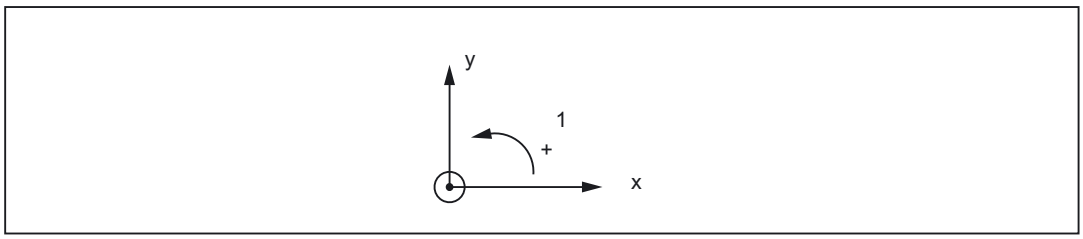
In this case, "t" is substituted by the number of the TRANSMIT transformations declared in the transformation data blocks (t may not be greater than 2).

Direction of rotation

The direction of rotation of the rotary axis is specified by machine data as described in the following paragraph.

TRANSMIT_ROT_SIGN_IS_PLUS_t

If the rotary axis rotates in an anti-clockwise direction on the X-Y plane when viewed along the Z axis, then the machine axis must be set to 1, but otherwise to 0.



MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_t=1

In this case, "t" is substituted by the number of the TRANSMIT transformations declared in the transformation data blocks (t may not be greater than 2).

Position of tool zero

The position of the tool zero point is specified by machine data as described in the following paragraph.

TRANSMIT_BASE_TOOL_t

Machine data:

MD24920 $MC_TRANSMIT_BASE_TOOL_t

is used to inform the control system of the position of the tool zero point in relation to the origin of the coordinate system declared for TRANSMIT. The machine data has three components for the three axes of the Cartesian coordinate system.

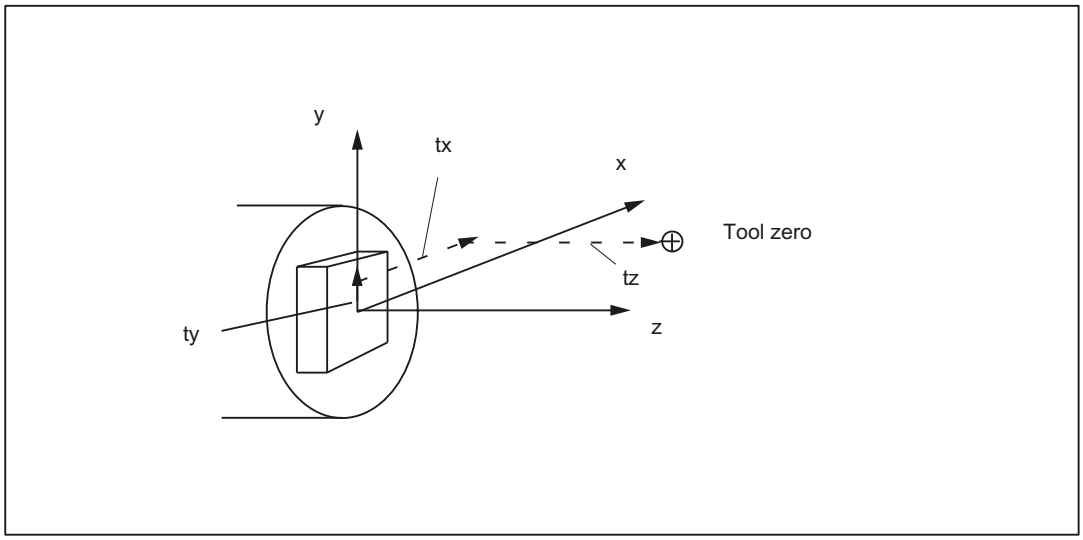


Figure 7-3 Position of tool zero in relation to origin of the Cartesian coordinate system

MD24920 $MC_TRANSMIT_BASE_TOOL_t[0]=tx

MD24920 $MC_TRANSMIT_BASE_TOOL_t [1]=ty

MD24920 $MC_TRANSMIT_BASE_TOOL_t [2]=tz

In this case, "t" in front of the index specification [ ] is substituted by the number of the TRANSMIT transformations declared in the transformation data blocks (t may not be greater than 2).

Replaceable geometry axes

The PLC is informed when a geometry axis has been replaced using GEOAX( ) through the optional output of an M code that can be set in machine data.

- MD22534 $MC_TRAFO_CHANGE_M_CODE

  Number of the M code that is output at the VDI interface in the case of transformation changeover.

Note

If this machine data is set to one of the values 0 to 6, 17, 30, then no M code is output.

References:

/FB1/ Function Manual Basic Functions; K2, "Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset"

7.2.3 Activation of TRANSMIT

TRANSMIT

After the settings described above have been made, the TRANSMIT function can be activated:

TRANSMIT or

TRANSMIT (t)

The first declared TRANSMIT function is activated with TRANSMIT. TRANSMIT(t) activates the t-th declared TRANSMIT function – t may not be greater than 2.

From software version 4 upwards, special procedures for pole transition etc. are also available with activation in accordance with "Machining Options for TRANSMIT".

Between activation of the function and deactivation as described below, the traversing movements for the axes of the Cartesian coordinate system can be programmed.

7.2.4 Deactivation of the TRANSMIT function

TRAFOOF

The keyword TRAFOOF deactivates an active transformation. When the transformation is deactivated, the base coordinate system is again identical to the machine coordinate system.

An active TRANSMIT transformation is likewise deactivated if one of the other transformations is activated.

(e.g. TRACYL, TRAANG, TRAORI).

References:

/FB3/ Function Manual Special Functions; "3-5 Axis Transformation"(F2).

7.2.5 Special system reactions with TRANSMIT

The transformation can be selected and deselected via parts program or MDA.

Please note on selection

- An intermediate motion block is not inserted (phases/radii).
- A series of spline blocks must be concluded.
- Tool radius compensation must be deselected.
- An activated tool length compensation is incorporated into the transformation in the geometry axis by the control
- The frame which was active prior to TRANSMIT is deselected by the control system. (corresponds to Reset programmed frame G500).

- An active working area limitation is deselected by the control for the axes affected by the transformation (corresponds to programmed WALIMOF).
- Continuous path control and rounding are interrupted.
- DRF offsets in transformed axes must have been deleted by the operator.

Please note on deselection

- An intermediate motion block is not inserted (phases/radii).
- A series of spline blocks must be concluded.
- Tool radius compensation must be deselected.
- The frame which was active prior to TRANSMIT is deselected by the control system. (Corresponds to Reset programmed frame G500).
- Continuous path control and rounding are interrupted.
- DRF offsets in transformed axes must have been deleted by the operator.
- Tool length compensation in the virtual axis (Y axis in the figure) is not executed.

Restrictions when TRANSMIT is active

The restrictions listed below have to be observed for an activated TRANSMIT function.

Tool change

Tools may only be changed when the tool radius compensation function is deselected.

Frame

All instructions which refer exclusively to the base coordinate system are permissible (FRAME, tool radius compensation). Unlike the procedure for inactive transformation, however, a frame change with G91 (incremental dimension) is not specially treated. The increment to be traversed is evaluated in the workpiece coordinate system of the new frame – regardless of which frame was effective in the previous block.

Rotary axis

The rotary axis cannot be programmed because it is occupied by a geometry axis and cannot thus be programmed directly as a channel axis.

Extensions

An offset in the rotary axis CM can be entered, for example, by compensating the inclined position of a workpiece in a frame within the frame chain. The x and y values are then as illustrated in the following diagram.

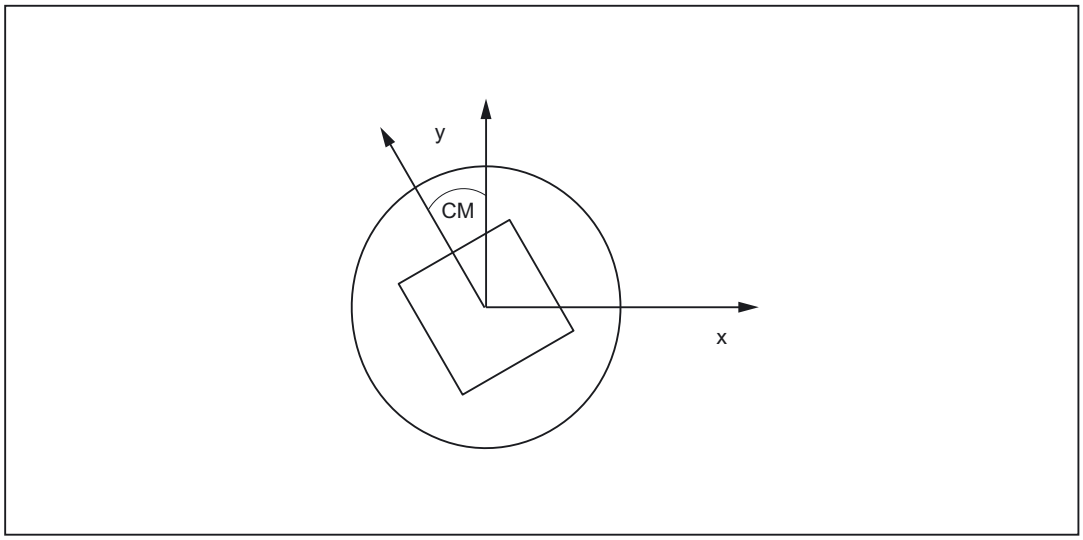


Figure 7-4 Rotary axis offset with TRANSMIT

This offset can also be included in the transformation as an offset in the rotary axis. To ensure that the total axial frame of the transmit rotary axis, i.e. the translation, fine offset, mirroring and scaling, is included in the transformation, the following settings must be made:

MD24905 $MC_TRANSMIT_ROT_AX_FRAME_1 = 1

MD24955 $MC_TRANSMIT_ROT_AX_FRAME_2 = 1

Note

Changes in the axis assignments are converted every time the transformation is selected or deselected. For further information about axial offsets for rotary axes to the SZS, see:

References:

/FB1/ Function Manual Basic Functions; "Co-ordinate Systems, Frames" (K2).

Pole

Movements through the pole (origin of Cartesian coordinate system) are disabled, i.e. a movement which traverses the pole is stopped in the pole followed by the output of an alarm. In the case of a cutter center offset, the movement is terminated accordingly at the end of the non-approachable area.

Options for pole transition and machining in pole vicinity are described in the section "Machining Options of TRANSMIT".

Exceptions

Axes affected by the transformation cannot be used

- as a preset axis (alarm)
- to approach the fixed point (alarm)
- for referencing (alarm)

Velocity control

The velocity monitoring function for `TRANSMIT` is implemented by default during preprocessing. Monitoring and limitation in the main run are activated:

- In AUTOMATIC mode if a positioning or oscillation axis has been programmed which is included in the transformation via machine data $MC_TRAFO_AXES_IN_n index 0 or 1.
- On changeover to JOG mode

The monitoring function is transferred from the main run back to the preprocessing routine if the axes relevant to the transformation process are operated as path axes.

The velocity monitoring function in preprocessing utilizes the machine better than the monitoring in the main run. Furthermore, the main run monitoring function deactivates the Look Ahead.

Interrupt parts program

If parts program processing is interrupted for JOG, then the following must be noted:

JOG

When JOG is selected, the conventional on-line velocity check is activated instead of the optimized velocity check.

From AUTOMATIC to JOG

If parts program processing is interrupted when the transformation is active followed by traversal in JOG mode, then the following must be noted when AUTOMATIC is selected again:

- The transformation is active in the approach block from the current position to the point of interruption. No monitoring for collisions takes place.

⚠ WARNING

The operator is responsible for ensuring that the tool can be re-positioned without any difficulties.

In AUTOMATIC mode

The velocity-optimized velocity planning function remains active for as long as the axes relevant to the transformation are traversed in mutual synchronism as path axes. If an axis involved in the transformation is traversed as a positioning axis, the online velocity check remains active until the transformation is deactivated or until all axes involved in the transformation are operating as path axes again. The return to velocity-optimized operation automatically initiates a STOPRE and synchronizes acyclic block pre-processing with the interpolation routine.

From start to reset

If parts program processing is aborted with RESET and restarted with START, then the following must be noted:

- The remaining parts program is traversed reproducibly only if all axes are traversed to a defined position by means of a linear block (G0 or G1) at the beginning of the parts program. A tool which was active on RESET may no longer be taken into account by the control (settable via machine data).

Power On, RESET

System response after Power On is determined by the settings stored in the following machine data:

MD20110 $MC_RESET_MODE_MASK and

MD20140 $MC_RAFO_RESET_VALUE

References:

/FB1/ Function Manual Basic Functions; "Workpiece-related Actual Value System" (K2).

Reference point approach

Axes cannot be referenced when a transformation is active. Any active transformation is deselected by the control system during a referencing operation.

7.2.6 Machining options for TRANSMIT

Introduction

The transformation TRANSMIT has a pole at the zero point of the TRANSMIT plane (example, see figure: 2–1, x = 0, Y = 0). The pole is located on the intersection between the radial linear axis and the rotary axis (X and CM). In the vicinity of the pole, small positional changes in the geometry axes generally result in large changes in position in the machine rotary axis. The only exceptions are linear motions into or through the pole.

A tool center point path through the pole does not cause the parts program to be aborted. There are no restrictions with respect to programmable traversing commands or active tool radius compensations. Nevertheless, workpiece machining operations close to the pole are not recommended since these may require sharp feedrate reductions to prevent overloading of the rotary axis.

New features

Definition:

A pole is said to exist if the line described by the tool center point intersects the turning center of the rotary axis.

The following cases are covered:

- Under what conditions and by what methods the pole can be traversed
- The response in pole vicinity
- The response with respect to working area limitations
- Monitoring of rotary axis rotations over 360°.

Pole traversal

The pole can be traversed by two methods:

- Traversal along linear axis
- Traversal into pole with rotation of rotary axis in pole

Traversal along linear axis

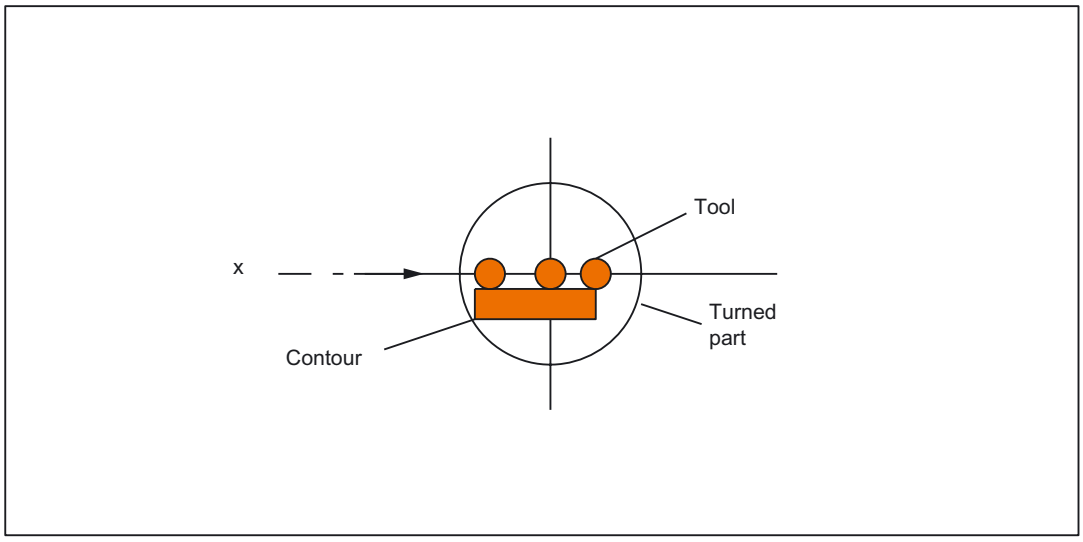


Figure 7-5 Traversal of x axis through pole

Rotation in pole

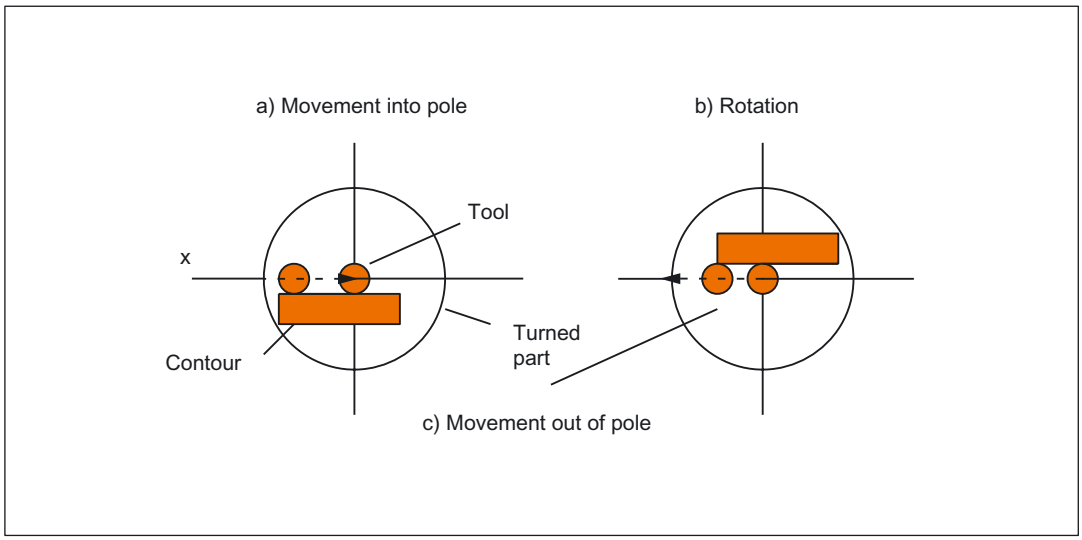


Figure 7-6 Traversal of x axis into pole (a), rotation (b), exit from pole (c)

Selection of method

The method must be selected according to the capabilities of the machine and the requirements of the part to be machined. The method is selected by machine data:

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2

The first MD applies to the first TRANSMIT transformation in the channel and the second MD correspondingly to the second TRANSMIT transformation.

VALUE	Meaning
0	Pole traversal The tool center point path (linear axis) must traverse the pole on a continuous path.
1	Rotation around the pole. The tool center point path must be restricted to a positive traversing range of the linear axis (in front of turning center).
2	Rotation around the pole. The tool center point path must be restricted to a negative traversing range of the linear axis (behind turning center).

Special features relating to pole traversal

The method of pole traversal along the linear axis may be applied in the AUTOMATIC and JOG modes.

System response:

Table 7- 1 Traversal of pole along the linear axis

Operating mode	State	Response
AUTOMATIC	All axes involved in the transformation are moved synchronously. TRANSMIT active.	High-speed pole traversal
	Not all axes involved in the transformation are traversed synchronously (e.g. position axis). TRANSMIT not active	Traversal of pole at creep speed
	An applied DRF (external zero offset) does not interfere with the operation. Servo errors may occur close to the pole during application of a DRF.	Abortion of machining, alarm
JOG	-	Traversal of pole at creep speed

Special features relating to rotation in pole

Precondition: This method is only effective in the AUTOMATIC mode.

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1 or 2

Value: 1 Linear axis remains within positive traversing range

Value: 2 Linear axis remains within negative traversing range

In the case of a contour that would require the pole to be traversed along the tool center point path, the following three steps are taken to prevent the linear axis from traversing in ranges beyond the turning center:

Step	Action
1	Linear axis traverses into pole
2	Rotary axis turns through 180°, the other axes involved in the transformation remain stationary.
3	Execution of remaining block. The linear axis now exits from the pole again.

In JOG mode, the motion stops in the pole. In this mode, the axis may exit from the pole only along the path tangent on which it approached the pole. All other motion instructions would require a step change in the rotary axis position or a large machine motion in the cases of minimum motion instructions. They are rejected with alarm 21619.

Traversal close to pole

If a tool center point traverses past the pole, the control system automatically reduces the feedrate and path acceleration rate such that the settings of the machine axes (MD 32000 $MA_MAX_AX_VELO[AX*] and MD32300 $MA_ MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the pole, the greater the reduction in the feedrate.

Tool center point path with corner in pole

A tool center point path which includes a corner in the pole will not only cause a step change in axis velocities, but also a step change in the rotary axis position. These cannot be reduced by decelerating.

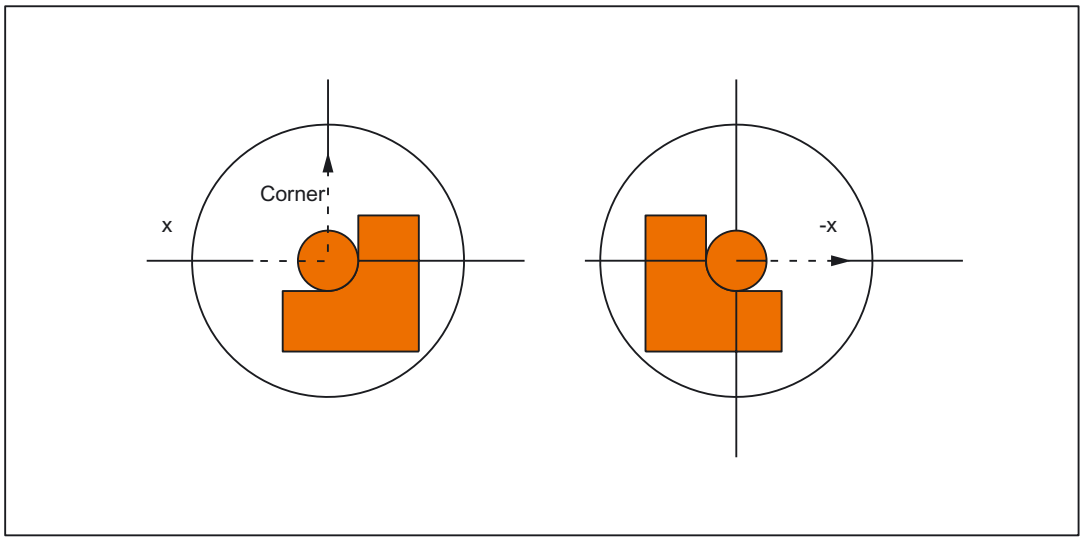


Figure 7-7 Pole traversal

Requirements:

AUTOMATIC mode,

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 0

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 0

The control system inserts a traversing block at the step change point. This block generates the **smallest possible rotation** to allow machining of the contour to continue.

Corner without pole traversal

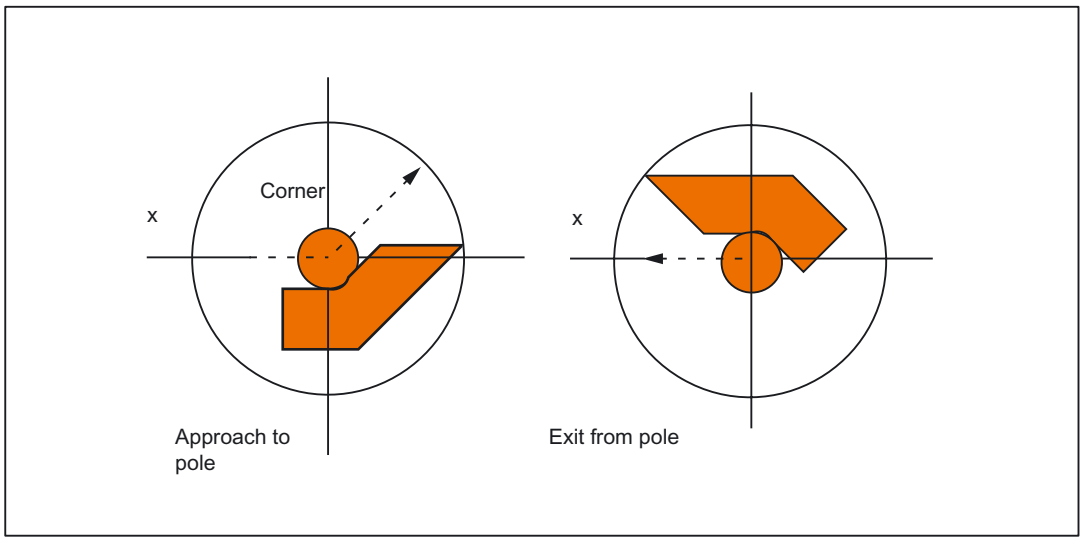


Figure 7-8 Machining on one pole side

Requirements:

AUTOMATIC mode,

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 or 2

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1 or 2

The control system inserts a traversing block at the step change point. This block generates the **necessary rotation** so that machining of the contour can continue on the same side of the pole.

Transformation selection in pole

If the machining operation must continue from a position on the tool center path which corresponds to the pole of the activated transformation, then an exit from the pole is specified for the new transformation.

If

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 0

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 0

is set (pole transition), then a rotation **as small as possible** is generated at the beginning of the block originating in the pole. Depending on this rotation, the axis then traverses either in front of or behind the turning center.

For

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 1

machining is done **before** the rotational center point (linear axis in positive traversing range), for

MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 2

or

MD24951 $MC_TRANSMIT_POLE_SIDE_FIX_2 = 2

behind the rotational center point (linear axis in negative traversing range).

Transformation selection outside pole

The control system moves the axes involved in the transformation without evaluating machine data MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_t. In this case, t = 1 stands for the first and t = 2 for the second TRANSMIT transformation in the channel.

7.2.7 Working area limitations

Starting position

When `TRANSMIT` is active, the pole is replaced by a working area limitation if the tool center point cannot be positioned in the turning center of the rotary axis involved in the transformation. This is the case when the axis perpendicular to the rotary axis (allowing for tool offset) is not positioned on the same radial plane as the rotary axis or if both axes are positioned mutually at an oblique angle. The distance between the two axes defines a cylindrical space in the BCS in which the tool cannot be positioned.

The illegal range cannot be protected by the software limit switch monitoring function since the traversing range of the machine axes is not affected.

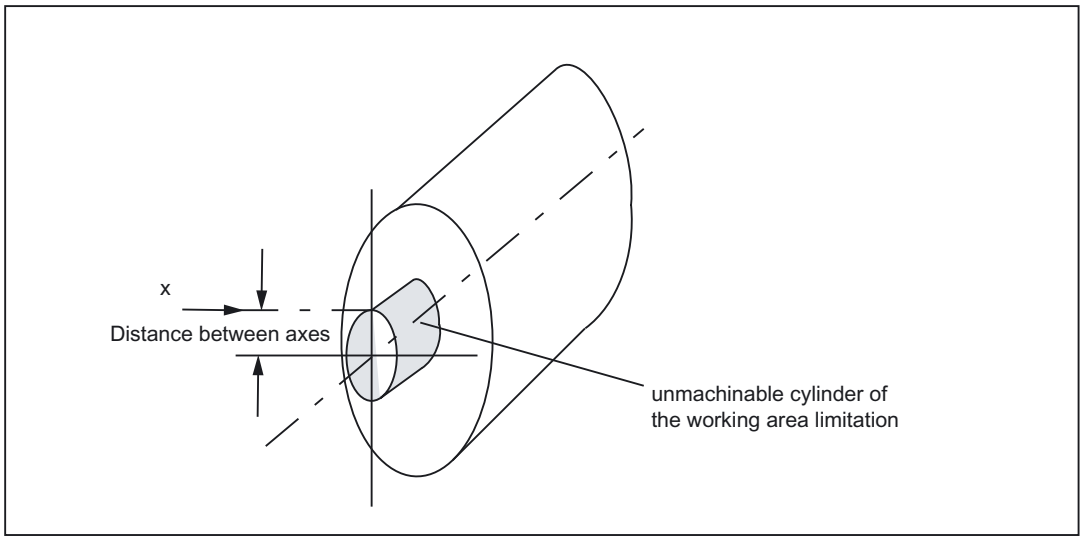


Figure 7-9 Working area limitation based on offset linear axis

Traverse into working area limitation

Any motion that leads into the working area limitation is rejected with alarm 21619. Any corresponding parts program block is not processed. The control system stops processing at the end of the preceding block.

If the motion cannot be foreseen promptly enough (JOG modes, positioning axes), then the control stops at the edge of the working area limitation.

Response close to working area limitation

If a tool center point traverses past the prohibited range, the control system automatically reduces the feedrate and path acceleration rate such that the settings of the machine axes (MD 32000 $MA_MAX_AX_VELO[AX*] and MD32300 $MA_ MAX_AX_ACCEL[AX*]) are not exceeded. The closer the path is to the working area limitation, the greater the reduction in the feedrate may be.

7.2.8 Overlaid motions with TRANSMIT

The control system cannot predict overlaid motions. However, these do not interfere with the function provided that they are very small (e.g. fine tool offset) in relation to the current distance from the pole (or from working area limitation). With respect to axes that are relevant for the transformation, the transformation monitors the overlaid motion and signals any critical quantity by alarm 21618. This alarm indicates that the block-related velocity planning function no longer adequately corresponds to the actual conditions on the machine. When the alarm is output, the conventional, non-optimized online velocity monitor is therefore activated. The preprocessing routine is re-synchronized with the main run by a `REORG` generated internally in the control.

Alarm 21618 should be avoided whenever possible since it indicates a state that can lead to axis overload and thus abortion of parts program processing.

7.2.9 Monitoring of rotary axis rotations over 360º

Ambiguity of rotary axis positions

The positions of the rotary axis are ambiguous with respect to the number of rotations. The control breaks down blocks containing several rotations around the pole into sub-blocks.

This subdivision must be noted with respect to parallel actions (e.g. output of auxiliary functions, block-synchronized positioning axis motions) since the programmed block end is no longer relevant for synchronization, but the end of the first sub-block. See:

References:

/FB1/ Function Manual, Basic Functions; "Auxiliary Functions to PLC" (H2)

/FBSY/ Function Manual, Synchronized Actions

In single block operation the control system machines individual blocks explicitly. In other modes, the sub-blocks are traversed with Look Ahead like a single block. A limitation of the rotary axis setting range is monitored by the software limit switch monitoring function.

7.2.10 Constraints

Look Ahead

All functions requiring Look Ahead (traversal through pole, Look Ahead) work satisfactorily only if the relevant axis motions can be calculated exactly in advance. With `TRANSMIT`, this applies to the rotary axis and the linear axis perpendicular to it. If one of these axes is the positioning axis, then the Look Ahead function is deactivated by alarm 10912 and the conventional online velocity check activated instead.

Selection of method

The **user is responsible** for making the optimum choice of "Traversal through pole" or "Rotation around pole".

Several pole traversals

A block can traverse the pole any number of times (e.g. programming of a helix with several turns). The parts program block is subdivided into a corresponding number of sub-blocks. Analogously, blocks which rotate several times around the pole are likewise divided into sub-blocks.

Rotary axis as modulo axis

The rotary axis can be a modulo rotary axis. However, this is not a mandatory requirement as was the case in SW 2 and 3. The relevant restrictions applying in SW 2 and 3 have been eliminated in SW 4.

Rotary axis as spindle

If the rotary axis without transformation is used as a spindle, it must be switched to position-controlled mode with `SPOS` before the transformation is selected.

TRANSMIT with supplementary linear axis

With active TRANSMIT, the channel identifier of posBCS[ax[3]] must have another name in the parts program, like the geometry axes. If posBCS[ax[3]] is written only outside the TRANSMIT transformation, this restriction does not apply if the axis has been assigned to a geometry axis. With active TRANSMIT, no contour information is processed via ax[3].

REPOS

It is possible to reposition on the sub-blocks produced as a result of the extended TRANSMIT function in SW 4. In this case, the control uses the first sub-block that is closest to the repositioning point in the BCS.

Block search

In the case of block search with calculation, the block end point (of the last sub-block) is approached in cases where intermediate blocks have been generated as the result of the extended functionality in SW 4.

7.3 TRACYL (option)

Note

The TRACYL transformation described below requires that unique names are assigned to machine axes, channels and geometry axes when the transformation is active. See

MD10000 $MN_AXCONF_MACHAX_NAME_TAB,

MD20080 $MC_AXCONF_CHANAX_NAME_TAB,

MD20060 $MC_AXCONF_GEOAX_NAME_TAB.

Besides this, no unequivocal assignments exist.

Task specification

Groove machining, see diagram.

Axis Configuration (1)

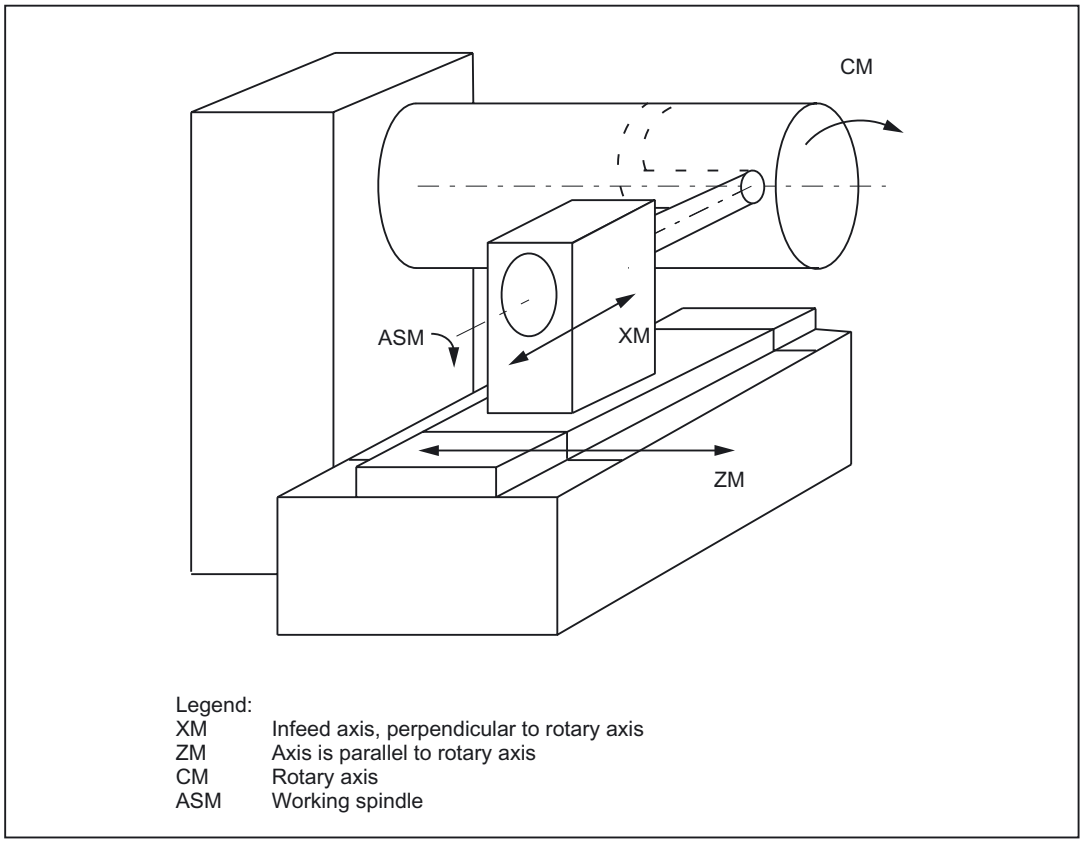


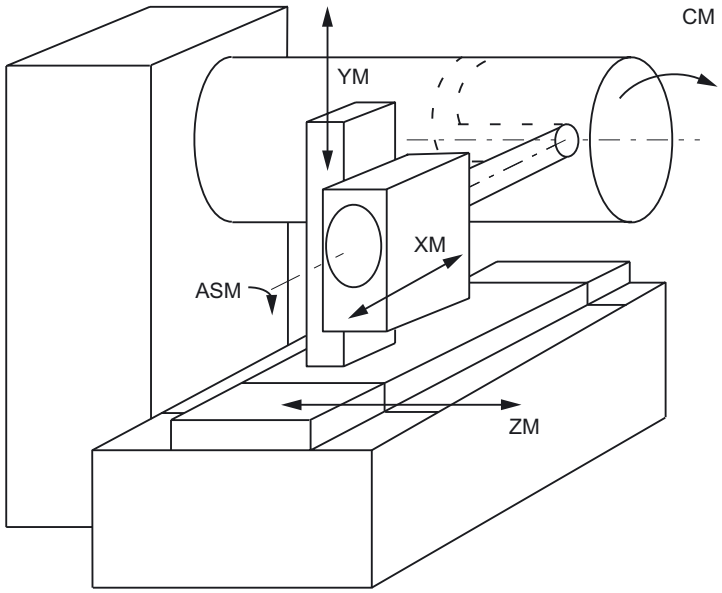
Figure 7-10 Machining grooves on a cylinder surface with X-C-Z kinematics

Axis Configuration (1)

The generated cylinder surface curve transformation allows a traversing path to be specified with respect to the generated surface of a cylinder coordinate system. The machine kinematics must correspond to the cylinder coordinate system. It must include one or two linear axes and a rotary axis. The two linear axes must be mutually perpendicular. The rotary axis must be aligned in parallel to one of the linear axes and intersect the second linear axis. In addition, the rotary axis must be co-linear to the cylinder coordinate system.

If there is only one linear axis (X), only grooves which are parallel to the periphery of the cylinder can be generated. In the case of two linear axes (X,Z), the groove pattern on the cylinder is optional.

Axis Configuration (2)



Legend:

XM	Infeed axis, perpendicular to rotary axis
YM	Additional axis
ZM	Axis is parallel to rotary axis
CM	Rotary axis
ASM	Working spindle

Figure 7-11 Machining grooves on a cylinder surface with X-Y-Z-C kinematics

If a third linear axis is available which can produce a right-handed Cartesian coordinate system with the other two linear axes (axis configuration 1), then it is used to offset the tool **parallel to the programmed path** by means of tool radius compensation. thereby allowing grooves with rectangular traversing section to be generated.

Functionality

During transformation (both axis configurations), the full functionality of the control is available, both for processing from the NC program and in JOG mode

Groove traversing-section

In the case of axis configuration 1, longitudinal grooves along the rotary axis are subject to parallel limits only if the groove width corresponds exactly to the tool radius.

Grooves in parallel to the periphery (transverse grooves) are not parallel at the beginning and end.

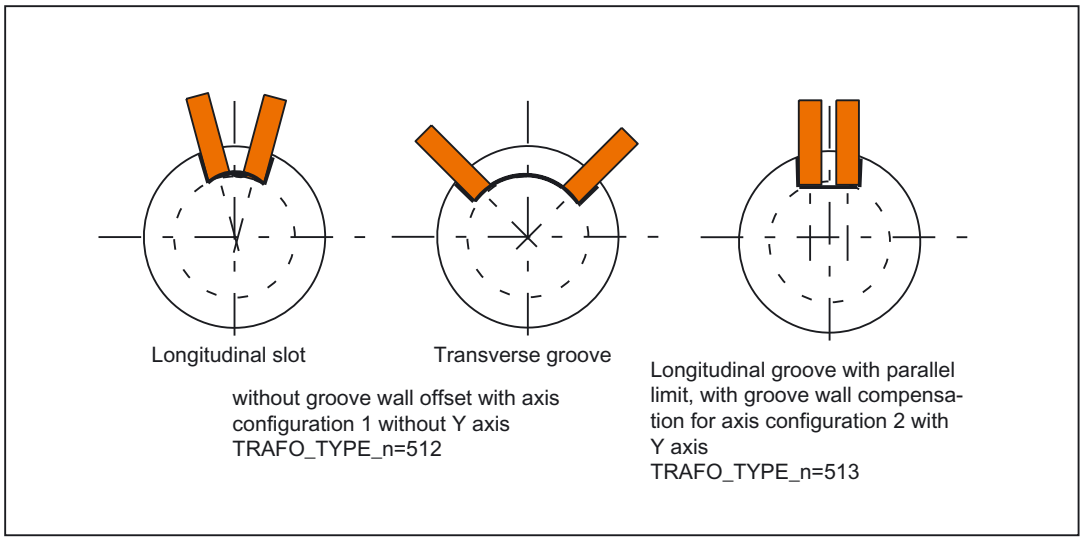


Figure 7-12 Grooves with and without groove wall offset

7.3.1 Preconditions for TRACYL

Number of transformations

Up to 10 transformation data blocks can be defined for each channel in the system. The machine data names of these transformations begin with $MC_TRAFO .. and end with ... _n, where n stands for a number between 1 and 10. The first machine data has the same meaning as described for TRANSMIT:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n

MD24100 $MC_TRAFO_TYPE_n

MD24110 $MC_TRAFO_AXES_IN_n.

The special settings described below apply to MD24100 $MC_TRAFO_TYPE_n and MD24100 $MC_TRAFO_AXES_IN_n with respect to generated cylinder surface transformation (TRACYL).

Number of TRACYL structures

Three of the 10 permitted data structures for transformations may be assigned to the TRACYL function. They are characterized by the fact that the value assigned with MD24100 $MC_TRAFO_TYPE_n is 512 or 513 or 514.

The following machine data must be set for a maximum of 3 of these TRACYL transformations:

MD24800 $MC_TRACYL_ROT_AX_OFFSET_t

MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_t

MD24820 $MC_TRACYL_BASE_TOOL_t

In this case, t specifies the number of the declared TRACYL transformation (maximum of 3).

Axis configuration

The following overview shows the relationship between the axes of the machine and the relevant axis data.

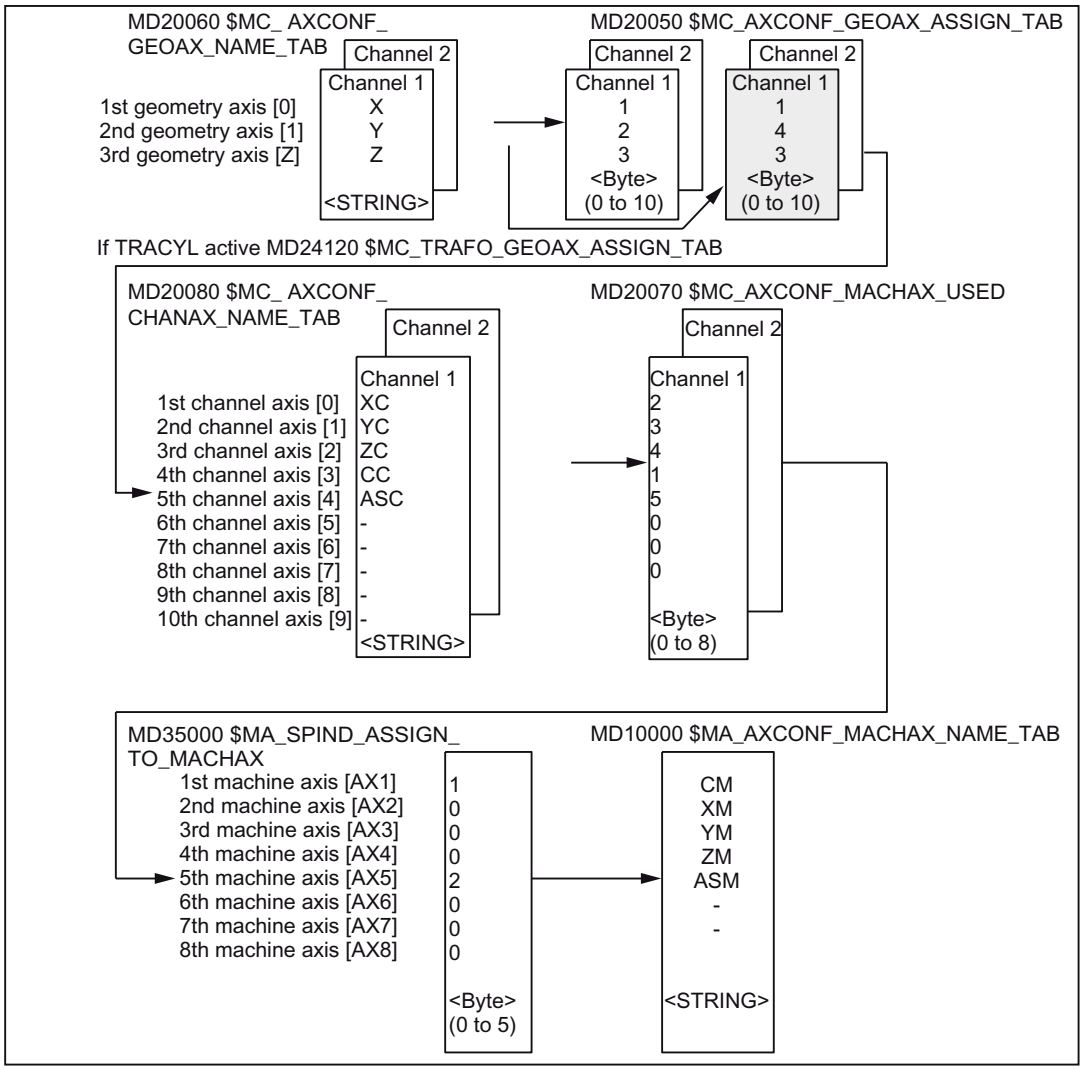


Figure 7-13 Axis configuration for the example in Figure "Machining grooves on a cylinder surface with X-Y-Z-C kinematics"

The configurations highlighted in the figure above apply when `TRACYL` is active.

Assignment of names to geometry axes

According to the above axis configuration overview, the geometry axes to be involved in the TRACYL operation must be defined with:

MD20050 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"

MD20050 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"

MD20050 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z",

for example, (the choice of names in the above figure is in accordance with defaults).

Assignment of geometry axes to channel axes

A distinction has to be made, whether`TRACYL` is active or not:

- TRACYL not active

  A Y axis is operated normally.

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0]=1

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2

  MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

- TRACYL active

  The Y axis becomes an axis in the direction of the cylinder co-ordinate system.

  MD20050 $MC_TRAFO_GEOAX_ASSIGN_TAB[0]=1

  MD20050 $MC_TRAFO_GEOAX_ASSIGN_TAB[1]=4

  MD20050 $MC_TRAFO_GEOAX_ASSIGN_TAB[2]=3

Entry of channel axes

Those axes, which do not belong to the Cartesian coordinate system, are entered.

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="YC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="ZC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="CC"

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4]="ASC"

Assignment of channel axes to machine axes

With the cd of the channel axes as a reference, the machine axis number to which the channel axes have been assigned, is transferred to the control system.

MD20070 $MC_AXCONF_MACHAX_USED[0]=2

MD20070 $MC_AXCONF_MACHAX_USED[1]=3

MD20070 $MC_AXCONF_MACHAX_USED[2]=4

MD20070 $MC_AXCONF_MACHAX_USED[3]=1

MD20070 $MC_AXCONF_MACHAX_USED[4]=5

(Entries in accordance with Figure "Machining grooves on a cylinder surface with X-Y-Z-C kinematics")

Identification of spindles

It is specified per machine axis, whether a spindle is present (value > 0: spindle number) or a path axis (value 0).

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[0]=1

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[1]=0

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[2]=0

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[3]=0

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[4]=2

Assignment of names to machine axes

With the cd of the machine axes as a reference, a machine axis name is transferred to the control system:

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="YM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ZM"

MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]="ASM"

7.3.2 Settings specific to TRACYL

Type of transformation

The following paragraph describes how the transformation type is specified.

TRAFO_TYPE_n

The user must specify the transformation type for the transformation data blocks (maximum n = 10). For TRACYL a VALUE of 512 must be set for axis configuration 1 and a value of 513 for axis configuration 2 or 514 for no groove side offset with supplementary linear axis. Transformation type 514 can also be activated with groove side offset by means of an additional parameter. See the chapter "Activation".

Example for VALUE 512: MD24100 $MC_TRAFO_TYPE_1=512

The setting must be made before TRACYL(d,t) is called – "t" is the number of the declared TRACYL transformation.

The TRACYL transformation requires only a rotary axis and a linear axis positioned perpendicular to the rotary axis. A real Y axis is used with transformation type 514 in order to compensate for a tool offset, for instance.

Transformation type 514 without groove side offset

Cylinder surface curve transformation TRAFO_TYPE_n = 514

If the machine has another linear axis which is perpendicular to both the rotary axis and the first linear axis, transformation type 514 can be used to apply tool offsets with the real Y axis. In this case, it is assumed that the user memory of the second linear axis is small and will not be used to execute the part program.

The existing settings for MD10000 $MC_TRAFO_GEOAX_ASSIGN_TAB_n apply.

Grooves with groove side offset

The required inclusion of the tool offset has already been taken into account for the TRACYL transformation with groove side offset.

Axis image

The following paragraph describes how the transformation axis image is specified.

TRAFO_AXES_IN_n

Three (or 4) channel axis numbers must be specified for TRACYL:

MD24110 $MC_TRAFO_AXES_IN_1[0]=channel axis number of the axis radial to the rotary axis.

MD24110 $MC_TRAFO_AXES_IN_1[1]=channel axis number of the rotary axis.

MD24110 $MC_TRAFO_AXES_IN_1[2]=channel axis number of the axis parallel to the rotary axis.

MD24110 $MC_TRAFO_AXES_IN_1[3]=channel axis number of the supplementary axis parallel to the cylinder surface and perpendicular to the rotary axis (provided two axis configurations are present).

Example in accordance with Figure "Machining grooves on a cylinder surface with X-Y-Z-C kinematics":

MD24110 $MC_TRAFO_AXES_IN_1[0]=1

MD24110 $MC_TRAFO_AXES_IN_1[1]=4

MD24110 $MC_TRAFO_AXES_IN_1[2]=3

MD24110 $MC_TRAFO_AXES_IN_1[3]=2

The setting must be made before TRACYL(d) or TRACYL(d,t) is activated. The axis numbers must refer to the channel axis sequences defined by the following machine data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n

Grooves without groove wall offset

For transformation type 514 the following indices apply for $MC_TRAFO_AXES_IN_n[ ].

Meaning of indices in relation to base coordinate system (BCS):

- [0]: Cartesian axis radial to rotary axis (if configured)
- [1]: Axis in generated cylinder surface perpendicular to rotary axis
- [2]: Cartesian axis parallel to rotary axis
- [3]: Linear axis parallel to index 2 in initial position of machine

Meaning of indices in relation to machine coordinate system (MCS):

- [0]: Linear axis radial to rotary axis (if configured)
- [1]: Rotary axis
- [2]: Linear axis parallel to rotary axis
- [3]: Linear axis perpendicular to the axes of indices [0] and [1]

Rotational position

The rotational position of the axis on the cylinder peripheral surface perpendicular to the rotary axis must be defined as follows:

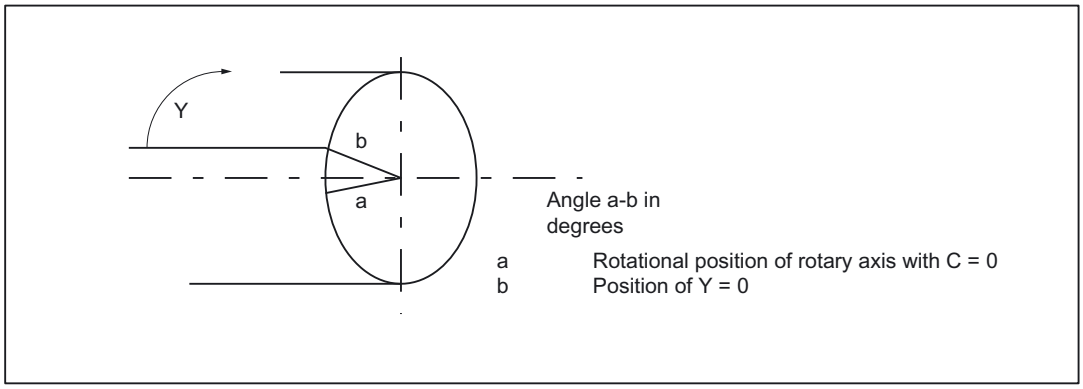


Figure 7-14 Center of rotation of axis in the peripheral cylinder surface

TRACYL_ROT_AX_OFFSET_t

The rotational position of the peripheral surface in relation to the defined zero position of the rotary axis is specified with:

MD24800 $MC_TRACYL_ROT_AX_OFFSET_t=...°

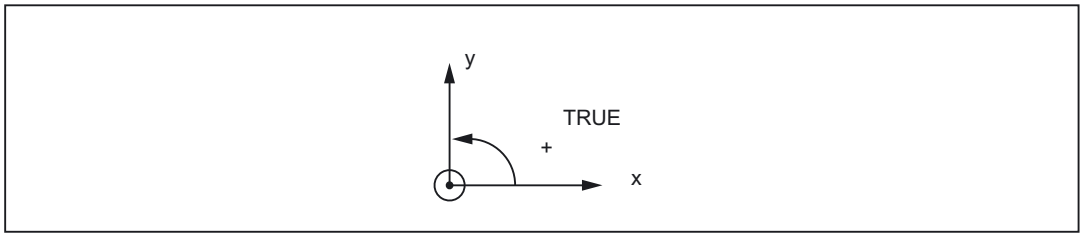
In this case, "t" is substituted by the number of the TRACYL transformations declared in the transformation data blocks (t may not be greater than 2).

Direction of rotation

The direction of rotation of the rotary axis is specified by machine data as described in the following paragraph.

TRACYL_ROT_SIGN_IS_PLUS_t

If the direction of rotation of the rotary axis on the x-y plane is counter-clockwise when viewed against the z axis, then the machine data must be set to TRUE, otherwise to FALSE.



MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_t=TRUE

In this case, "t" is substituted by the number of the `TRACYL`transformations declared in the transformation data blocks (t may not be greater than 2).

Replaceable geometry axes

The PLC is informed when a geometry axis has been replaced using GEOAX( ) through the optional output of an M code that can be set in machine data.

- MD22534 $MC_TRAFO_CHANGE_M_CODE

  Number of the M code that is output at the VDI interface in the case of transformation changeover.

Note

If this machine data is set to one of the values 0 to 6, 17, 30, then no M code is output.

References:

/FB1/ Function Manual Basic Functions; Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset (K2)

Position of tool zero

The position of the tool zero point in relation to the origin of the Cartesian coordinate system is specified by machine data as described in the following paragraph.

TRACYL_BASE_TOOL_t

MD24820 $MC_TRACYL_BASE_TOOL_t

The above machine data is used to inform the control of the position of the tool zero point in relation to the origin of the cylinder coordinate system declared for TRACYL. The machine data has three components for the axes X, Y, Z of the machine coordinate system.

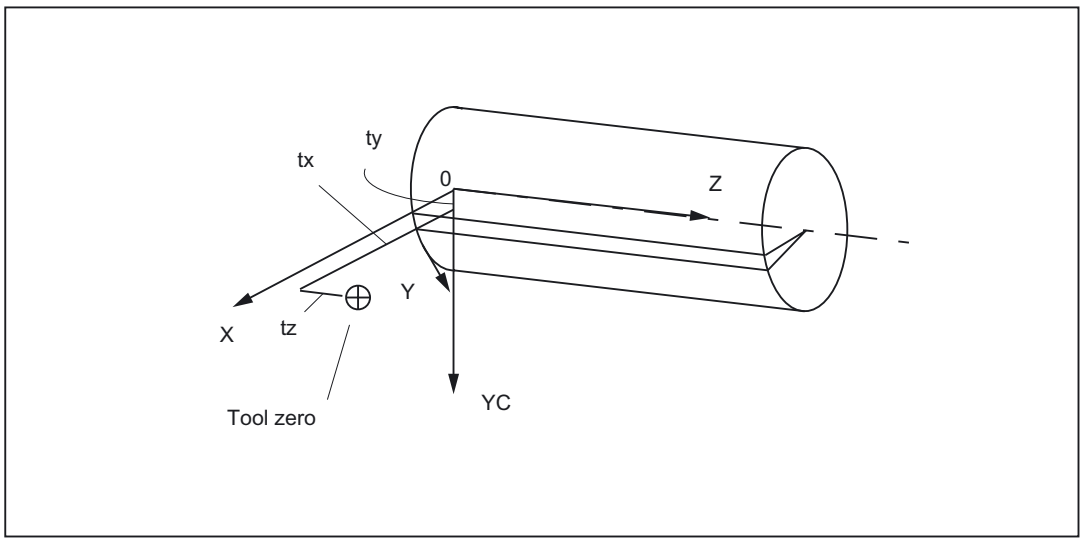


Figure 7-15 Position of tool zero in relation to machine zero

Example:

MD24820 $MC_TRACYL_BASE_TOOL_t[0]=tx

MD24820 $MC_TRACYL_BASE_TOOL_t[1]=ty

MD24820 $MC_TRACYL_BASE_TOOL_t[2]=tz

In this case, "t" is substituted by the number of the TRACYL transformations declared in the transformation data blocks (t may not be greater than 2).

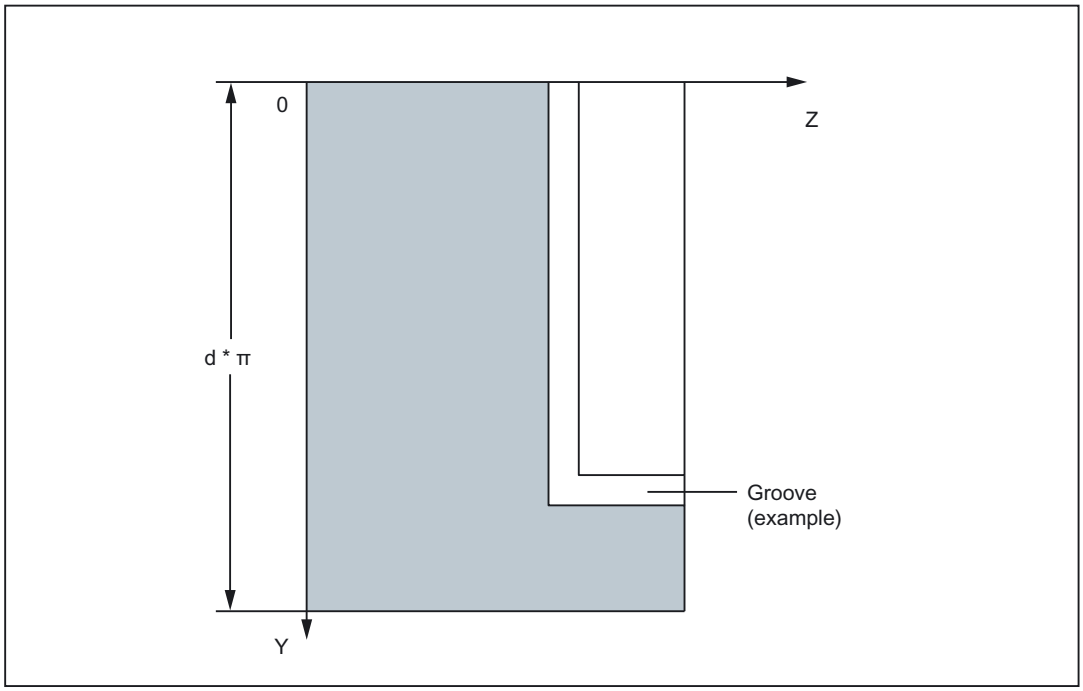


Figure 7-16 Cylinder coordinate system

7.3.3 Activation of TRACYL

TRACYL

After the settings described above have been made, the TRACYL function can be activated:

TRACYL(d)

or

TRACYL(d,t) TRACYL(reference diameter, Tracyl data block)

TRACYL(d) is used to activate the first declared TRACYL function. TRACYL(d,t) activates the t-th declared TRACYL function – t may not be greater than 2. The value d stands for the current diameter of the cylinder to be machined.

Between activation of the function and deactivation as described below, the traversing movements for the axes of the cylinder coordinate system can be programmed.

Transformation type 514 with groove side offset

An additional call parameter is used for transformation type 514; this is the third parameter with which TRACYL transformation with groove side offset can be selected:

TRACYL(reference diameter, Tracyl data block, groove side offset).

- Reference diameter: Obligatory parameter (must always be defined)

  Range of values: >0
- Tracyl data block: Optional parameter, preset value is 1

  Range of values: 1,2
- Groove side offset: Optional parameter, preset value corresponds to value specified in machine data

  MD24808 $MC_TRACYL_DEFAULT_MODE_1) or

  MD24858 $MC_TRACYL_DEFAULT_MODE_2)

  Range of values: 0,1

7.3.4 Deactivation of the TRACYL function

TRAFOOF

The keyword TRAFOOF deactivates an active transformation. When the transformation is deactivated, the base coordinate system is again identical to the machine coordinate system.

An active TRACYL transformation is likewise deactivated if one of the other transformations is activated in the relevant channel (e.g., TRANSMIT, TRAANG, TRAORI).

References:

/FB3/ Function Manual Special Functions; 5- Axis Transformation (F2)

7.3.5 Special system reactions with TRACYL

The transformation can be selected and deselected via parts program or MDA.

Please note on selection

- An intermediate motion block is not inserted (phases/radii).
- A series of spline blocks must be concluded.
- Tool radius compensation must be deselected.
- The frame which was active prior to TRACYL is deselected by the control system.

  (corresponds to Reset programmed frame G500).

- The control system deselects an active working area limit for axes affected by the transformation.

  (Corresponds to programmed WALIMOF).
- Continuous path control and rounding are interrupted.
- DRF offsets must have been deleted by the operator.
- In the case of cylinder generated surface curve transformation with groove wall compensation (axis configuration 2, TRAFO_TYPE_n=513), the axis used for the correction (TRAFO_AXES_IN_n[3]) must be set to zero (y=0) so that the groove is machined in the center of the programmed groove center line.

Please note on de-selection

The same points apply as for selection.

Restrictions when TRACYL is active

The restrictions listed below must be noted when the TRACYL function is active:

Tool change

Tools may only be changed when the tool radius compensation function is deselected.

Supplementary conditions for TRACYL without groove side offset

With active TRANSMIT, the channel identifier of posBCS[ax[3]] must have another name in the parts program, like the geometry axes. If posBCS[ax[3]] is written only outside the TRACYL transformation, this restriction does not apply if the axis has been assigned to a geometry axis. With active TRACYL, no contour information is processed via ax[3].

Frame

All instructions which refer exclusively to the base coordinate system are permissible (FRAME, tool radius compensation). Unlike the procedure for inactive transformation, however, a frame change with G91 (incremental dimension) is not specially treated. The increment to be traversed is evaluated in the workpiece coordinate system of the new frame - regardless of which frame was effective in the previous block.

Rotary axis

The rotary axis cannot be programmed because it is occupied by a geometry axis and cannot thus be programmed directly as a channel axis.

Extensions

An offset of the rotary axis CM can be entered, for example, by compensating the inclined position of a workpiece in a frame within the frame chain. The x and y values are then as illustrated in the following diagram.

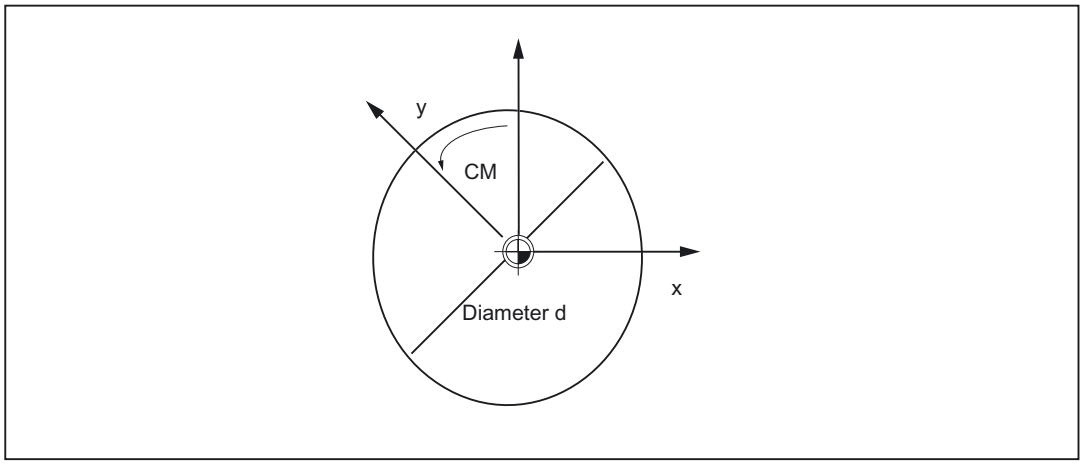


Figure 7-17 Rotary axis offset with TRACYL

This offset can also be included in the transformation as an offset in the rotary axis or as y-offset. To ensure that the total axial frame of the TRACYL rotary axis, i.e. the translation, fine offset, mirroring and scaling, is included in the transformation, the following settings must be made:

MD24805 $MC_TRACYL_ROT_AX_FRAME_1 = 1

MD24855 $MC_TRACYL_ROT_AX_FRAME_2 = 1

Note

Changes in the axis assignments are converted every time the transformation is selected or deselected. For further information about axial offsets for rotary axes to the SZS as an offset on the peripheral surface, please see:

References:

/FB1/ Function Manual Basic Functions; Co-ordinate Systems, Frames (K2)

Axis utilization

The axes:

- in the generated cylinder surface perpendicular to the rotary axis (Y) and
- additional axis (YC)

may not be used as a positioning or oscillation axis.

Exceptions

Axes affected by the transformation cannot be used

- as a preset axis (alarm)
- to approach the fixed point (alarm)
- for referencing (alarm)

Interrupt parts program

The following points must be noted with respect to interrupting parts program processing in connection with TRACYL:

AUTOMATIC after JOG

If parts program processing is interrupted when the transformation is active followed by traversal in JOG mode, then the following must be noted when AUTOMATIC is selected again:

- The transformation is active in the approach block from the current position to the point of interruption. No monitoring for collisions takes place.

⚠ WARNING
The operator is responsible for ensuring that the tool can be re-positioned without any difficulties.

START after RESET

If parts program processing is aborted with `RESET` RESET and restarted with `START`, then the following must be noted:

- The remaining parts program is traversed reproducibly only if all axes are traversed to a defined position by means of a linear block (G0 or G1) at the beginning of the parts program. A tool which was active on `RESET` may no longer be taken into account by the control (settable via machine data).

7.3.6 Jog

Special features of JOG

When generated cylinder surface transformation with groove wall compensation ($MC_TRAFO_TYPE = 513) is active in JOG mode, it must be noted that the axes are traversed depending on the preceding status in `AUTOMATIC`. When groove wall compensation is active, the axes movement therefore differs from the situation when the correction function is deselected. The parts program can therefore be continued (`REPOS`) after a parts program interruption.

7.4 TRAANG (option)

Note

The TRAANG transformation described below requires that unique names are assigned to machine axes, channels and geometry axes when the transformation is active. See

MD10000 $MN_AXCONF_MACHAX_NAME_TAB,

MD20080 $MC_AXCONF_CHANAX_NAME_TAB,

MD20060 $MC_AXCONF_GEOAX_NAME_TAB.

Besides this, no unequivocal assignments exist.

Task specification

Grinding operations

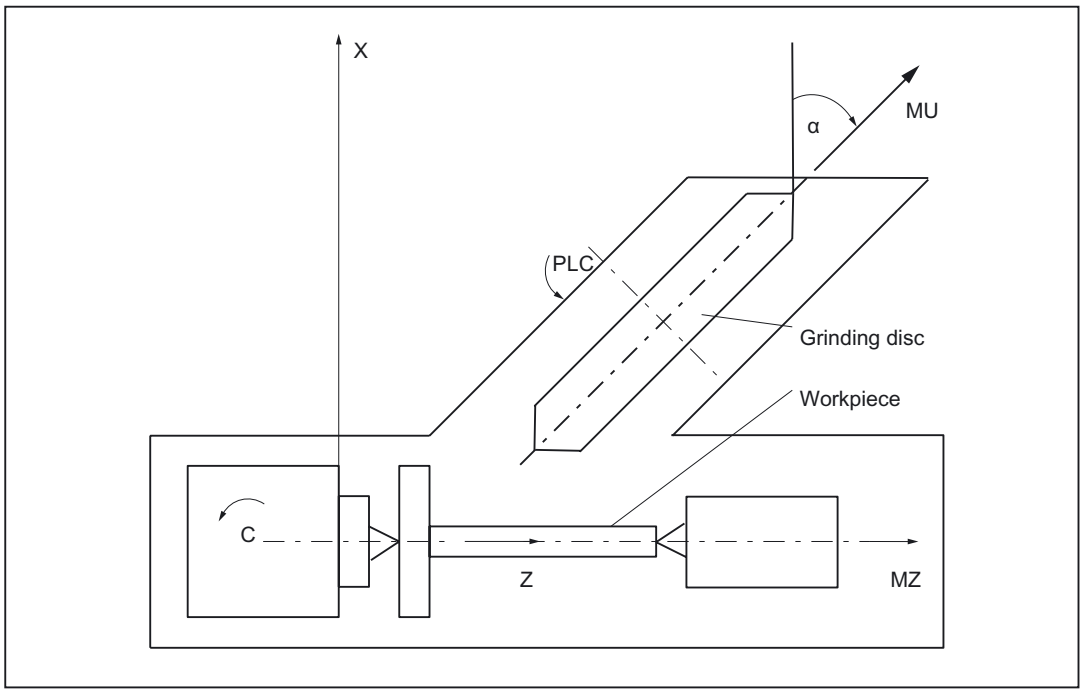


Figure 7-18 Machine with inclined infeed axis

Legend:

X, Z: Cartesian coordinate system for programming

C: Rotary axis

AS: Working spindle

MZ: Machine axis (linear)

MU: Inclined axis

The following range of machining operations is available:

- Longitudinal grinding
- Face grinding
- Grinding of a specific contour
- Oblique plunge-cut grinding

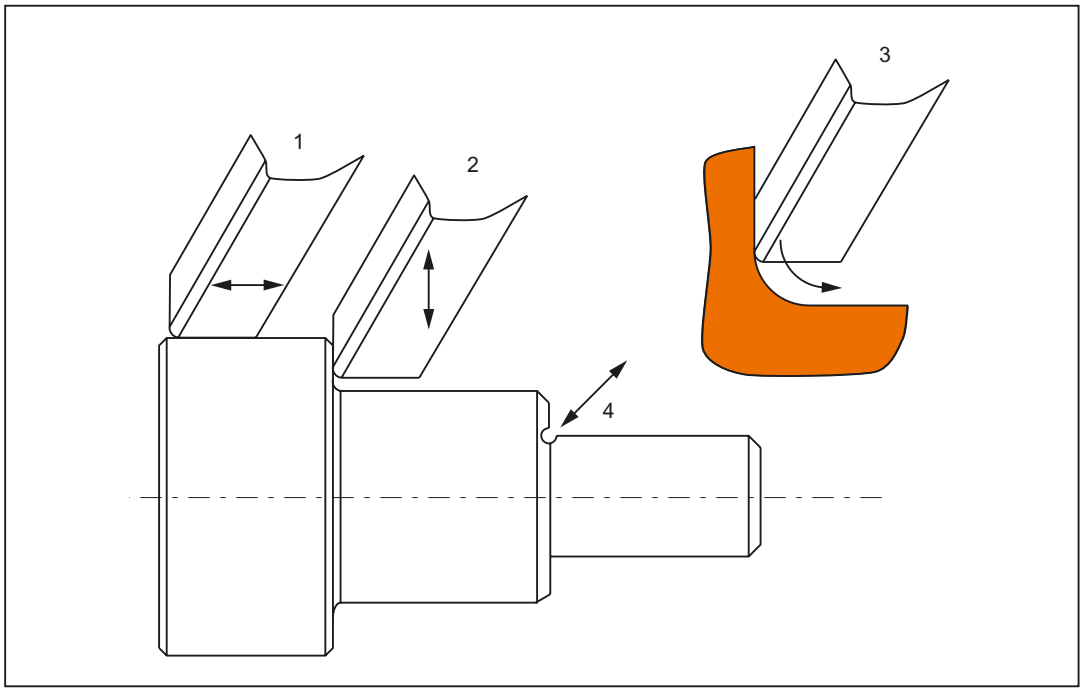


Figure 7-19 Possible grinding operations

7.4.1 Preconditions for TRAANG (inclined axis)

Axis configuration

To be able to program in the Cartesian coordinate system (see figure "Machine with inclined infeed axis": X, Y, Z), it is necessary to inform the control of the correlation between this coordinate system and the actually existing machine axes (MU,MZ):

- Assignment of names to geometry axes
- Assignment of geometry axes to channel axes
  - general situation (inclined axis not active)
  - inclined axis active
- Assignment of channel axes to machine axis numbers
- Identification of spindles
- Allocation of machine axis names.

With the exception of "Inclined axis active", the procedure is the same as for the normal axis configuration.

References:

/FB1/ Function Manual Basic Functions; Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset (K2).

Number of transformations

Up to 10 transformation data blocks can be defined for each channel in the system. The machine data names of these transformations begin with $MC_TRAFO .. and end with ... _n, where n stands for a number between 1 and 10. The following sections include descriptions of these data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_n

MD24100 $MC_TRAFO_TYPE_n

MD24110 $MC_TRAFO_AXES_IN_n

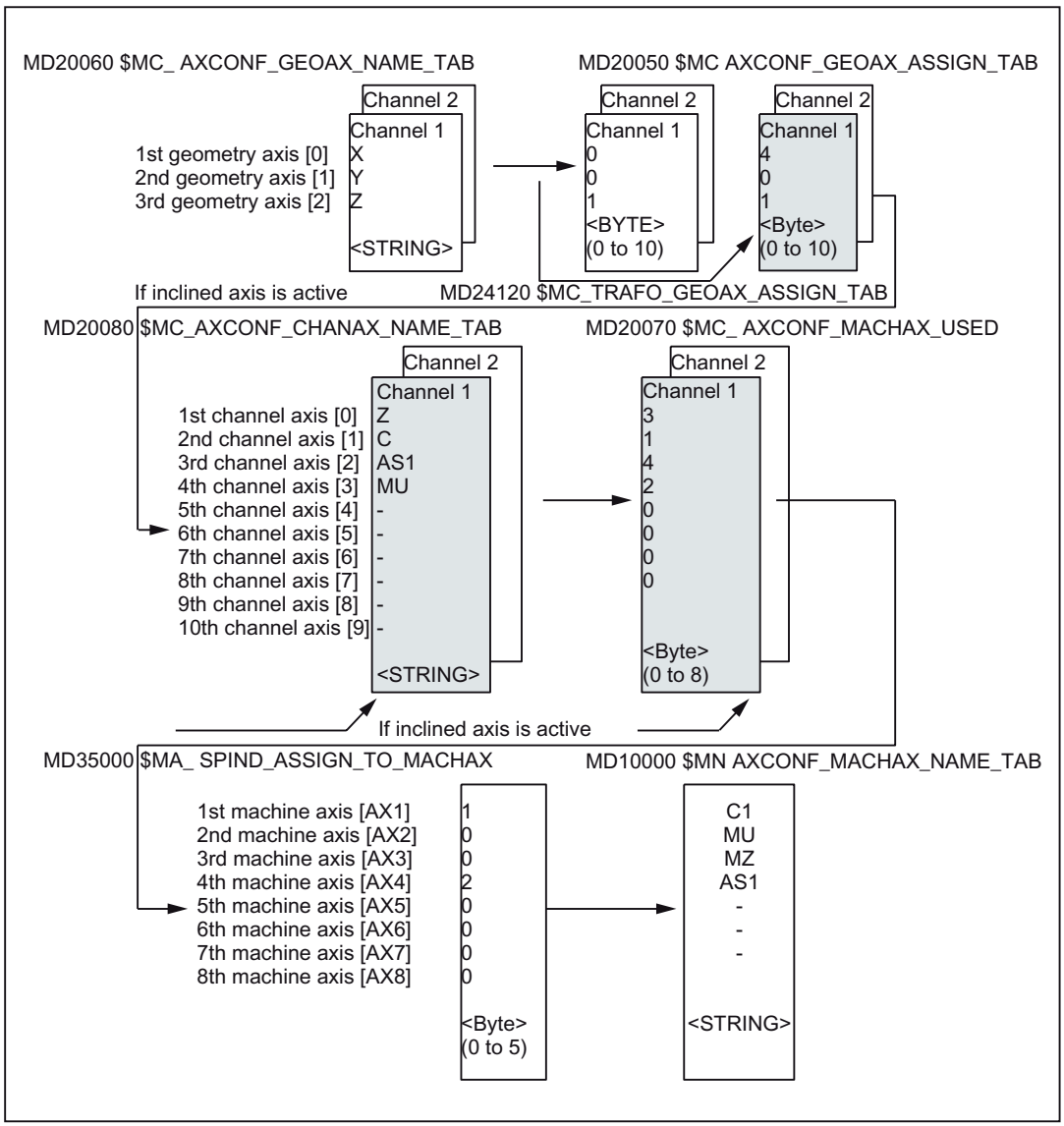
Number of inclined axes

Two of the 10 permitted data structures for transformations may be assigned to the inclined axis function. They are characterized by the fact that the

value assigned to MD24100 $MC_TRAFO_TYPE_n is 1024.

Axis configuration

The axes of the grinding machine illustrated in the figure, must be entered as follows in the machine data:



Axis configuration for the example in figure "Machine with inclined infeed axis"

The configurations highlighted in the figure above apply when `TRAANG` is active.

7.4.2 Settings specific to TRAANG

Type of transformation

TRAFO_TYPE_n

The user must specify the transformation type for the transformation data blocks (maximum n = 10) in the following machine data:

MD24100 $MC_TRAFO_TYPE_n

The value for an inclined axis is 1024:

MD24100 $MC_TRAFO_TYPE_1=1024

Axis image

TRAFO_AXES_IN_n

Two channel axis numbers must be specified for the transformation data block n:

MD24110 $MC_TRAFO_AXES_IN_1[0]=4; channel axis number of the inclined axis

MD24110 $MC_TRAFO_AXES_IN_1[1]=1; channel axis number of the axis parallel to Z

MD24110 $MC_TRAFO_AXES_IN_1[2]=0; channel axis number not active

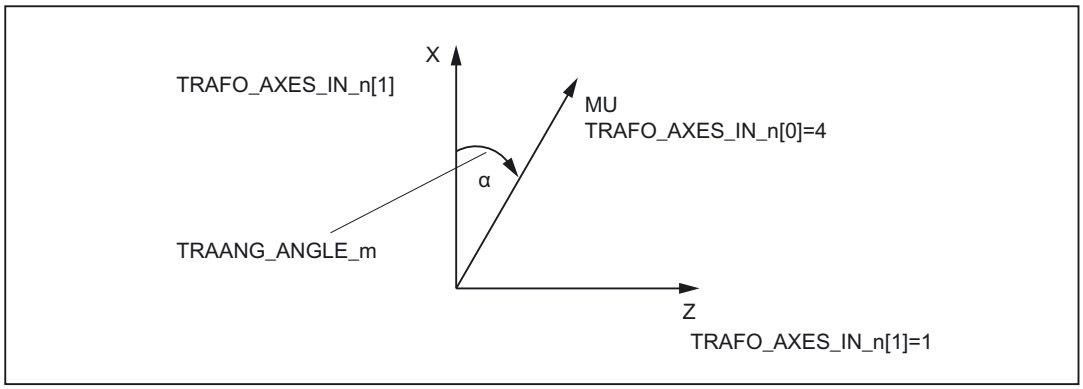


Figure 7-20 Parameter TRAANG_ANGLE_m

Assignment of geometry axes to channel axes

Example:

MD24430 $MC_TRAFO_TYPE_5 = 8192 chaining

MD24110 $MC_TRAFO_AXIS_IN_1[0..x]

MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1 Definition geometry axis assignment of Trafo 1

MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =6 Definition geometry axis assignment of Trafo 1

MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3 Definition geometry axis assignment of Trafo 1

MD24996 $MC_TRACON_CHAIN_2[0] = 2 input variables in TRACON

MD24996 $MC_TRACON_CHAIN_2[1] = 3 input variables in TRACON

MD24996 $MC_TRACON_CHAIN_2[2] = 0 input variables in TRACON

MD24996 $MC_TRACON_CHAIN_2[3] = 0 input variables in TRACON

Angle of inclined axis

TRAANG_ANGLE_m

The following machine data is used to inform the control system of the angle which exists between a machine axis and the inclined axis in degrees:

MD24700 $MC_TRAANG_ANGLE_m

MD24700 $MC_TRAANG_ANGLE_m = angle between a Cartesian axis and the associated inclined machine axis in degrees. The angle is counted positively in the clockwise direction (see figure "Machine with inclined infeed axis", angle α).

In this case, "m" is substituted by the number of the TRAANG transformation declared in the transformation data blocks – m may not be greater than 2.

Permissible angular range

The permissible angular range is:

-90° < TRAANG_ANGLE_m < 0°

0° < TRAANG_ANGLE_m < 90°

No transformation is required for 0°.

For +/- 90° the inclined axis is parallel to the second linear axis.

Position of tool zero

TRAANG_BASE_TOOL_m

The following machine data is used to inform the control of the position of the tool zero point in relation to the origin of the coordinate system declared for the inclined axis function:

MD24710 $MC_TRAANG_BASE_TOOL_m

The machine data has three components for the three axes of the Cartesian coordinate system.

Zero is entered as default.

The corrections are not converted when the angle is changed.

Optimization of velocity control

The following machine data are used to optimize the velocity control in jog mode and in positioning and oscillation modes:

TRAANG_PARALLEL_VELO_RES_m

Machine data MD24720 $MC_TRAANG_PARALLEL_VELO_RES_m is used to set the velocity reserve which is held on the parallel axis for compensatory motion (see the following machine data).

MD24110 $MC_TRAFO_AXES_IN_n[1]

Range of values: 0 ... 1

0: When value 0 is set, the control system automatically determines the reserve: the axes are limited with equal priority (= default setting).

>0: Values >0 result in the reserve to be set as allowed machine axis velocity for the parallel axis in the following machine data:

MD24720 $MC_TRAANG_PARALLEL_VELO_RES_m

The velocity characteristics of the vertical axis are determined by the control system on the basis of the reserve.

TRAANG_PARALLEL_ACCEL_RES_m

The following machine data is used to set the axis acceleration reserve which is held ready on the parallel axis (see MD24110 $MC_TRAFO_AXES_IN_n[1]) for compensatory motion:

MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_m

Range of values: 0 ... 1

0: When value 0 is set, the control system automatically determines the reserve: the axes are accelerated with equal priority. (= default)

>0: Values >0 result in the acceleration to be set as allowed machine acceleration value for the parallel axis in the following machine data:

MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_m

The velocity characteristics of the vertical axis are determined by the control system on the basis of the reserve.

Replaceable geometry axes

The PLC is informed when a geometry axis has been replaced using GEOAX( ) through the optional output of an M code that can be set in machine data.

- MD22534 $MC_TRAFO_CHANGE_M_CODE

  Number of the M code that is output at the VDI interface in the case of transformation changeover.

  Note

  If this machine data is set to one of the values 0 to 6, 17, 30, then no M code is output.

References:

/FB1/ Function Manual Basic Functions; Coordinate Systems, Axis Types, Axis Configurations, Workpiece-related Actual-Value System, External Zero Offset (K2).

7.4.3 Activation of TRAANG

TRAANG(a)

After the settings described above have been made, the TRAANG function can be activated:

TRAANG(a)

or

TRAANG(a,n)

With TRAANG(a) the first declared transformation inclined axis is activated.

The angle of the inclined axis can be specified with α.

- If α is omitted or a zero is entered, the transformation is activated with the parameterization of the previous selection.

  On the first selection, the presettings according to the machine data apply.
- If α (angle) is omitted (e.g. TRAANG(), TRAANG(,n)), the transformation is activated with the parameterization of the previous selection. On the first selection, the presettings according to the machine data apply. An angle α = 0 (e.g.TRAANG(0), TRAANG(0,n)) is a valid parameter setting and is no longer equivalent to the omission of the parameter, as in the case of older versions. The permissible value range for a is: -90 degrees < a < + 90 degrees.

  TRAANG(a,n) activates the nth declared inclined axis transformation.

  This form is required only if several transformations are activated in the channel – n may not be greater than 2.

Programming variants

TRAANG(a,1) == TRAANG(a,0) == TRAANG(a,) == TRAANG(a)

Between activation of the function and deactivation as described below, the traversing movements for the axes of the Cartesian coordinate system must be programmed.

7.4.4 Deactivation of TRAANG

TRAFOOF

The keyword TRAFOOF deactivates an active transformation. When the transformation is deactivated, the base coordinate system is again identical to the machine coordinate system.

An active TRAANG transformation is likewise deactivated if one of the other transformations is activated in the relevant channel (e.g., TRANSMIT, TRAORI).

References:

/FB3/ Function Manual Special Functions; 5- Axis Transformation (F2).

7.4.5 Special system reactions with TRAANG

The transformation can be selected and deselected via parts program or MDA.

Selection and deselection

- An intermediate motion block is not inserted (phases/radii).
- A spline block sequence must be terminated.
- Tool radius compensation must be deselected.
- The current frame is deselected by the control system.

  (corresponds to programmed G500).
- An active working area limitation is deselected by the control for the axes affected by the transformation (corresponds to programmed `WALIMOF`).
- An activated tool length compensation is included in the transformation by the control.
- Continuous path control and rounding are interrupted.
- DRF offsets must have been deleted by the operator.
- All axes specified in machine data MD24110 $MC_TRAFO_AXES_IN_n must be synchronized on a block-related basis (e.g. no traversing instruction with POSA...).

Restrictions

Tool change

Tools may only be changed when the tool radius compensation function is deselected.

Frame

All instructions which refer exclusively to the base coordinate system are permissible (`FRAME`, tool radius compensation). Unlike the procedure for inactive transformation, however, a frame change with G91 (incremental dimension) is not specially treated. The increment to be traversed is evaluated in the workpiece coordinate system of the new frame - regardless of which frame was effective in the previous block.

Extensions

When `TRAANG` is selected and deselected, the assignment between geometry axes and channel axes can change. The user can apply these geometric contour sections to the axial frame as a translation, rotation, scaling and mirroring in relation to the x and z plane with respect to the inclined infeed axis.

For additional information on these frame corrections with transformations, see:

References:

/FB1/ Function Manual Basic Functions; Axes, Coordinate Systems, Frames (K2)

Exceptions

Axes affected by the transformation cannot be used

- as a preset axis (alarm)
- to approach the fixed point (alarm)
- for referencing (alarm)

Velocity control

The velocity monitoring function for TRAANG is implemented as standard during preprocessing.

Monitoring and limitation in the main run are activated:

- in AUTOMATIC mode, if a positioning or oscillation axis has been programmed that is involved in the transformation.
- On changeover to JOG mode

The monitoring function is transferred again from the main run to block preprocessing if the preprocessing is re-synchronized with the main run (currently, for example, on changeover from JOG to AUTOMATIC).

The velocity monitoring function in preprocessing utilizes the dynamic limitations of the machine better than the monitoring function in the main run.

This also applies to machines on which, with oblique machining operations,

7.4.6 Inclined axis programming (G05, G07)

Function

The following functions are available:

- Position programming and display in the Cartesian coordinate system
- Cartesian calculation of tool offset and zero offset
- Programming of angles for the inclined axis in the NC program
- Approach starting position for inclined plunge cutting (G07)
- Inclined plunge cutting (G05)
- In JOG mode, the movement of the grinding wheel can either be cartesian or in the direction of the inclined axis (the display stays Cartesian).

  Selection is done via DB21-28 DBX29.4 "PTP travel". If PTP travel is activated, only the real U axis moves, the Z axis display is updated.

Programming

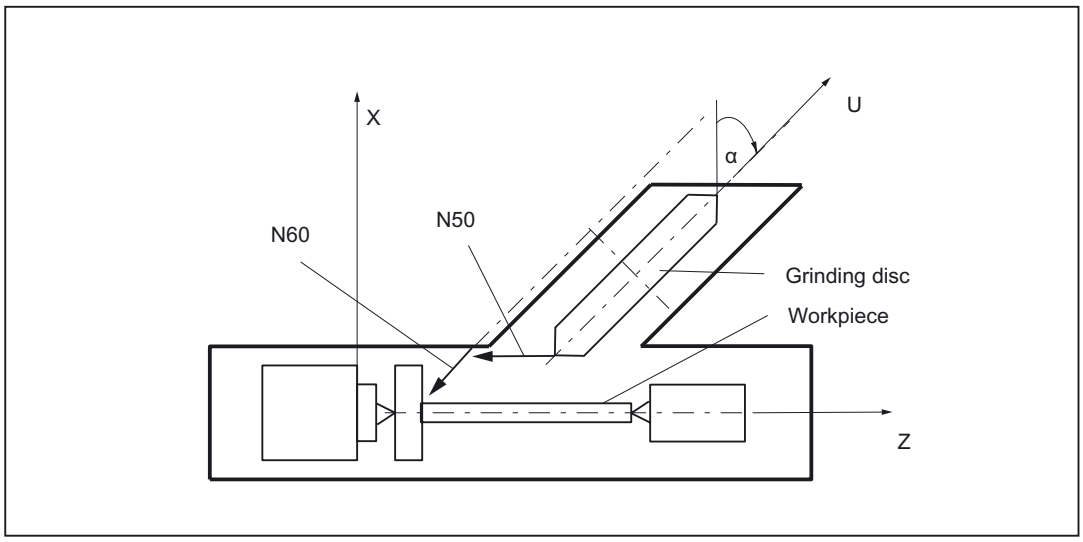


Figure 7-21 Machine with inclined infeed axis

Example:

```
N...                                    Program axis for inclined axis
N50 G07 X70 Z40 F4000                   Approach starting position
N60 G05 X70 F100                        Oblique plunge-cutting
N...
```

Constraints

- It is only meaningful to select the function "Cartesian PTP travel" in JOG mode (motion according to G05) if transformation is active (TRAANG). Note the value set in MD20140 $MC_TRAFO_RESET_VALUE.
- REPOS offsets must be traversed back in JOG mode in the Cartesian coordinates while "PTP travel" is not active.
- The Cartesian working area limitation is monitored for overtravel in jog mode if "PTP travel" is active and the axis will brake before overtraveling. If "PTP travel" is not active, the axis can be traversed right up to the operating range limit.

See also

Cartesian PTP travel (Page 561)

7.5 Chained transformations

Introduction

It is possible to chain the kinematic transformation described here, with an additional transformation of the type "Inclined axis":

- TRANSMIT
- TRACYL
- TRAANG (oblique axis)

as described in

References:

/FB3/ Function Manual Special Functions; 3- to 5-axis Transformation (F2).

- Orientation transformations
- Universal milling head

Applications

The following is a selection from the range of possible chained transformations:

- Grinding contours that are programmed as a side line of a cylinder (TRACYL) using an inclined grinding wheel, e.g., tool grinding.
- Finish cutting of a contour that is not round and was generated with TRANSMIT using an inclined grinding wheel.

Note

The transformations described below require that individual names are assigned to machine axes, channels and geometry axes when the transformation is active. Compare the following machine data:

MD10000: AXCONF_MACHAX_NAME_TAB

MD20080: AXCONF_CHANAX_NAME_TAB

MD20060: AXCONF_GEOAX_NAME_TAB

Besides this, no unequivocal assignments exist.

Axis configuration

The following configuration measures are necessary for a chained transformation:

- Assignment of names to geometry axes
- Assignment of names to channel axes
- Assignment of geometry axes to channel axes
  - general situation (no transformation active)
- Assignment of channel axes to machine axis numbers

- Identification of spindle, rotation, modulo for axes
- Allocation of machine axis names.
- Transformation-specific settings (for individual transformations **and** for **chained** transformations)
  - Transformation type
  - axes going into transformation
  - Assignment of geometry axes to channel axes during active transformation
  - depending on transformation, rotational position of the co-ordinate system, tool zero point in relation to the original co-ordinate system, angle of the inclined axis, etc.

Number of transformations

Up to ten transformation data blocks can be defined for each channel in the system. The machine data names of these transformations begin with $MC_TRAFO .. and end with ... _n, where n stands for a number between 1 and 10.

Number of chained transformations

Within the maximum of 10 transformations of a channel, a maximum of **two chained** transformations may be defined.

Transformation sequence

When configuring the machine data, the data concerning the single transformations (that may also become part of chained transformations) must be specified before the data concerning the chained transformations.

Chaining sequence

With chained transformations the second transformation must be "inclined axis" (`TRAANG`).

Chaining direction

The BCS is the input for the first of the transformations to be chained; the MCS is the output for the second one.

Constraints

The supplementary conditions and special cases indicated in the individual transformation descriptions are also applicable for use in chained transformations.

7.5.1 Activating chained transformations

TRACON

A chained transformation is activated by:

TRACON(trf, par)

- trf:

  Number of the chained transformation: 0 or 1 for first/only chained transformation. If nothing is programmed here, then this has the same meaning as specifying value 0 or 1, i.e., the first/only transformation is activated – 2 for the second chained transformation. (Values not equal to 0 - 2 generate an error alarm).

- par

  One or more parameters separated by a comma for the transformations in the chain expecting parameters, for example, the angle of the inclined axis. If parameters are not set, the defaults or the parameters last used take effect. Commas must be used to ensure that the specified parameters are evaluated in the sequence in which they are expected, if default settings are to be effective for previous parameters. In particular, a comma is required before at least one parameter, even though it is not necessary to specify trf. For example: TRACON(, 3.7).`TRACON`(, 3.7).

If another transformation was previously activated, it is implicitly disabled by means of `TRACON`().

7.5.2 Switching off a chained transformation

TRAFOOF

A chained transformation is switched off with `TRAFOOF` just like any other transformation.

7.5.3 Special characteristics of chained transformations

Tool data

A tool is always assigned to the first transformation in a chain. The subsequent transformation then behaves as if the active tool length were zero. Only the basic tool lengths set in the machine data (_BASE_TOOL_) are valid for the **first** transformation in the chain.

Example

The chapter "Chained transformations" contains configuration examples for single transformations and the transformation chains created from them.

See also

Chained transformations (Page 550)

7.5.4 Persistent transformation

Function

A persistent transformation is always active and has a relative effect to the other explicitly selected transformations. Other selected transformation are computed as the first chained transformation in relation to the persistent transformation.

Transformations such as TRANSMIT that must be selected in relation to the persistent transformation must be parameterized in a chain with the persistent transformation by means of TRACON. It is the first chained transformation rather than the TRACON transformation which is programmed in the parts program.

For additional programming tips see

References:

/PGA/ Programming Manual Job Planning; Transformations "Chained Transformation"

Selection and deselection

Persistent transformation is selected via the following machine data:

MD20144 $MC_TRAFO_MODE_MASK, Bit 0 = 1

MD20144 $MC_TRAFO_RESET_VALUE defines persistent transformation.

MD20140 $MC_TRAFO_RESET_VALUE=Number of the transformation data set of the persistent transformation

In addition the following must be set (i.e. noted):

MD20110 $MC_RESET_MODE_MASK

Bit 0 = 1 (Bit 7 is evaluated)

Bit 7 =0 (MD20140 $MC_TRAFO_RESET_VALUE determines the transformation data set)

MD20112 $MC_START_MODE_MASK (MD20140 $MC_TRAFO_RESET_VALUE)

MD20118 $MC_GEOAX_CHANGE_RESET= TRUE (i.e. geometry axes are reset).

If this additional data is not parameterized correctly,

alarm 14404 is generated.

With TRAFOOF the active TRACON is deselected and the persistent transformation is automatically selected.

Effects on HMI operation

As a transformation is always active with the persistent transformation, the HMI user interface is adapted accordingly for the selection and deselection of transformations:

`TRACON` on HMI

Accordingly the HMI operator interface does **not** display `TRACON`, but the first chain transformation of `TRACON` e.g. `TRANSMIT` . Accordingly, the transformation type of the 1st chained transformation is returned by the corresponding system variable, i.e. $P_TRAFO and $AC_TRAFO. Cycles written in `TRANSMIT` can then be used directly.

`TRACOOF` on HMI

In accordance with the `TRAFOOF` programming instruction **no** transformation is displayed in the G code list on the HMI user interface. System variables $P_TRAFO and $AC_TRAFO therefore return a value of 0, the persistent transformation is operative and the BCS and MCS coordinate systems do not coincide. The displayed MCS position always refers to the actual machine axes.

System variables

New system variables return the transformation types of the active chained transformations.

Description	NCK variable
no transformation active: 0 one transformation active: Type of 1st chained transformation with TRACON, or type of active transformation if not TRACON	$P_TRAFO_CHAIN[0]
no transformation active: 0 one transformation active: Type of 2nd chained transformation with TRACON	$P_TRAFO_CHAIN[1] $AC_TRAFO_CHAIN[1]
are only used if more than 2 transformations are chained. These variables presently only return 0.	$P_TRAFO_CHAIN[2] $AC_TRAFO_CHAIN[2] and $P_TRAFO_CHAIN[3] $AC_TRAFO_CHAIN[3]

Display persistent transformation:

$P_TRAFO_CHAIN[0], $AC_TRAFO_CHAI[0]

These settings allow an active transformation to be displayed reliably in the part program or in cycles.

Difference between a TRACON and the other transformations:

$P_TRAFO, $AC_TRAFO if no transformation is active, or $P_TRAFO_CHAIN[1], $AC_TRAFO_CHAI[1] is interrogated for a value other than zero.

Frames

Frame adjustments for selection and deselection of the TRACON are carried out as if there was only the first chained transformation. Transformations on the virtual axis cease to be effective when TRAANG is selected.

JOG

The persistent transformation remains in effect when traversing with JOG.

Constraints

The persistent transformation does not change the principle operating sequences in the NCK. All restrictions applying to an active transformation also apply to the persistent transformation.

A RESET command still deselects any active transformation completely; the persistent transformation is selected again. The persistent transformation is not reselected under error conditions. A corresponding alarm is generated to indicate the error constellation.

Alarm 14401 or 14404 can be activated when TRAANG is the persistent transformation. When the persistent transformation is active, other transformation alarms may generated in response to errors depending on the transformation type selected.

The transformation is deselected implicitly during referencing. A RESET or START command must be issued after referencing, in order to reselect the persistent transformation.

Example

For a lathe with an inclined additional Y axis, the transformation of the inclined axis should be part of the machine configuration and therefore does not have to be considered by the programmer. With TRACYL or TRANSMIT transformations are selected, which must then include the TRAANG. When the programmed transformations are deactivated, TRAANG is automatically activated again. In the HMI operator interface TRACYL or TRANSMIT is displayed.

Machine data for a turning machine with Y1 axis inclined in relation to X1 but perpendicular to Z1.

CANDATA (1)

; Kinematic without transformations

MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "Y2"

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3

; Data for TRAANG

MD24100 $MC_TRAFO_TYP_1 = 1024; TRAANG Y1 axis inclined to X1, perpendicular to Z1

MD24110 $MC_TRAFO_AXES_IN_1[0]=2

MD24110 $MC_TRAFO_AXES_IN_1[1]=1

MD24110 $MC_TRAFO_AXES_IN_1[2]=3

```
MD24110 $MC_TRAFO_AXES_IN_1[3] = 0
MD24110 $MC_TRAFO_AXES_IN_1[4] = 0
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3
MD24700 $MC_TRAANG_ANGLE_1 = 60
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
 ; Definition of persistent transformation
MD20144 $MC_TRAFO_MODE_MASK = 1
MD20140 $MC_TRAFO_RESET_VALVUE= 1
MD20110 $MC_RESET_MODE_MASK = 'H01'
MD20112 $MC_START_MODE_MASK = 'H80'
MD20140 $MC_TRAFO_RESET_VALUE
MD20118 $MC_GEOAX_CHANGE_RESET= TRUE
 ; Data for TRANSMIT, TRACYL
MD24911 $MC_TRANSMIT_POLE_SIDE_FIX_1 = 1 ; also 2, causes alarm 21617
MD24200 $MC_TRAFO_TYP_2 = 257
MD24210 $MC_TRAFO_AXES_IN_2[0] = 1
MD24210 $MC_TRAFO_AXES_IN_2[1] = 4
MD24210 $MC_TRAFO_AXES_IN_2[2] = 3
MD24210 $MC_TRAFO_AXES_IN_2[3] = 0
MD24210 $MC_TRAFO_AXES_IN_2[4] = 0
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =4
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3
MD24300 $MC_TRAFO_TYP_3 = 514
MD24310 $MC_TRAFO_AXES_IN_3[0] = 1
MD24310 $MC_TRAFO_AXES_IN_3[1] = 4
MD24310 $MC_TRAFO_AXES_IN_3[2] = 3
MD24310 $MC_TRAFO_AXES_IN_3[3] = 0
MD24310 $MC_TRAFO_AXES_IN_3[4] = 0
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =1
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =4
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =3
 ; Data for TRACON
```

```
 ; TRACON chaining TRANSMIT 514/TRAANG(Y1 axis inclined in relation to X1)
MD24400 $MC_TRAFO_TYP_4 = 8192
MD24995 $MC_TRACON_CHAIN_1[0] = 3
MD24995 $MC_TRACON_CHAIN_1[1] = 1
MD24995 $MC_TRACON_CHAIN_1[2] = 0
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =4
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
 ; TRACON chaining TRANSMIT 257/TRAANG(Y1 axis inclined in relation to X1)
MD24430 $MC_TRAFO_TYP_5 = 8192
MD24996 $MC-TRACON_CHAIN_2[0] = 2
MD24996 $MC-TRACON_CHAIN_2[1] = 1
MD24996 $MC_TRACON_CHAIN_2[2] = 0
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =4
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
M17
 ; matching parts program:
$TC_DP1[1,1]=120; Tool type
$TC_DP2[1,1] = 0
$TC_DP3[1,1]=3 ; length compensation vector
$TC_DP4[1,1]=25
$TC_DP5[1.1] =5
$TC_DP6[1,1]= 2; Radius; tool radius
 ; transformation change:
N1000 G0 X0 Y=0 Z0 A80 G603 SOFT G64
N1010 N1020 X10 Y20 Z30 ; TRAANG(,1) not possible, since automatically selected
N1110 TRANSMIT(1) N1120 X10 Y20 Z30N1130 Y2=0 ; TRACON(2) not necessary, since
translated automatically
N1210 TRAFOOF ; TRAANG(,1) not necessary, since translated automatically
N1220 X10 Y20 Z30
M30
```

7.5.5 Axis positions in the transformation chain

Function

System variables having the following content are provided for machines with system- or OEM transformations, especially for chained transformations (TRACON):

Type	System variable	Description
REAL	$AA_ITR[ax,n]	Current setpoint value at output of the nth transformation
REAL	$AA_IBC[ax]	Current setpoint value of a cartesian axis
REAL	$VA_ITR[ax,n]	Current actual value at output of the nth transformation
REAL	$VA_IBC[ax]	Current cartesian BCS encoder position of an axis
REAL	$VA_IW[ax]	Current WCS actual value of an axis
REAL	$VA_IB[ax]	Current BCS encoder position of an axis

The following must be observed with reference to the control system responses:

- POWER ON

  The encoder position has the value 0 for not-referenced axes. The encoder actual values are inverse-transformed accordingly for the $VA variables.

- RESET

  An active transformation can change in RESET, which has a direct influence on the values of the system variables. An active transformation which is active again after RESET, is deactivated for a short duration and then reactivated. This has a direct influence on the position variables. The values of variables can change.

  Via the variable:

  $AC_STAT == 0

  this status can be queried in synchronous actions.

$AA_ITR[<axis>, <transformer layer>]

The $AA_ITR[ax,n] variable determines the setpoint position of an axis at the output of the nth chained transformation.

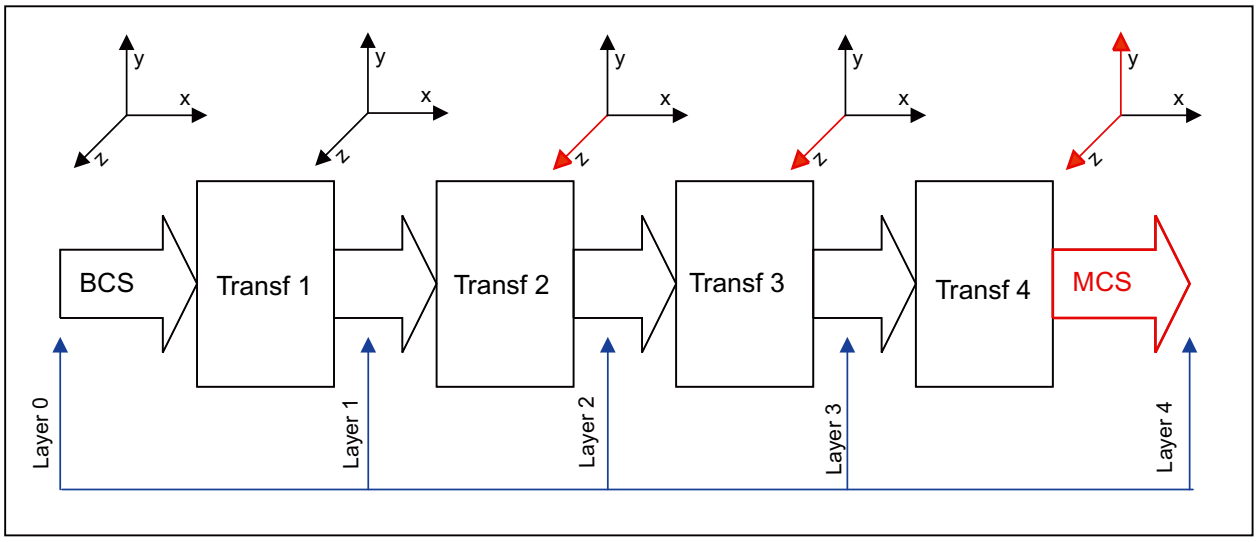


Figure 7-22 Transformer layer

Transformer layer

The 2nd index of the variable corresponds to the transformer layer in which the positions are tapped:

- Transformer layer 0: The positions correspond to the BCS positions, i.e.,:

  $AA_ITR[x,0] == $AA_IB[x]
- Transformer layer 1: Setpoint positions at output of 1st transformation
- Transformer layer 2: Setpoint positions at output of 2nd transformation
- Transformer layer 3: Setpoint positions at output of 3rd transformation
- Transformer layer 4: Setpoint positions at output of 4th transformation, i.e.,

  $AA_ITR[x,4] == $AA_IM[x]

If one or more transformations of the transformer chain are missing, the highest planes continue to deliver the same values. If, e.g., Transformer 3 and Transformer 4 are missing, this corresponds to:
$AA_ITR[x,2] = $AA_ITR[x,3] = $AA_ITR[x,4] = $AA_IM[x]

If the transformations are shut off via `TRANSFORMER OFF` or in RESET, the Layers 0 to 4 fuse and the variable always delivers the BCS value (Layer 0) in this case.

Axis

Either a geometrical-, channel- or a machine axis identifier is permissible as the 1st index of the variable. The assignment of the channel axes to the geometry axes corresponding to the 0 level takes place during the programming of geometry axis identifier in each transformer layer. Using geometry axis identifiers is meaningful only if the geometry axes are not switched over. Otherwise it is always better to use channel axis identifiers.

$AA_IBC[<axis>]

The variable $AA_IBC[ax] determines the setpoint position of a cartesian axis lying between BCS and MCS. If an axis is cartesian at the output of the nth transformation, then this output value is delivered. If the corresponding axis at the output of all transformations is not cartesian, then the BCS value including all BCS offsets of the axis are determined.

If TRACON responds to an axis as cartesian, then its MCS value is delivered. The used axis identifier can be a geometrical-, channel- or a machine axis identifier.

$VA_ITR[<axis>, <transformer layer>]

The variable $VA_ITR[ax,n] determines the encoder position of an axis at the output of the nth chained transformation.

$VA_IBC[<axis>]

The variable $VA_IBC[ax] determines the encoder position of a cartesian axis lying between BCS and MCS. The used axis identifier can be a geometrical-, channel- or a machine axis identifier.

If an axis at the output of the nth transformation is cartesian, then this output value is delivered. If the corresponding axis at the output of all transformations is not cartesian, then the BCS value of the axis is determined.

$VA_IW[<axis>]

The variable $VA_IW[ax] determines the encoder position of an inverse-transformed axis in WCS. The WCS value contains all axis-related superimposition portions (DRF, AA_OFF, ext. zero offset etc.) and offset values (CEC, etc.).

$VA_IB[<axis>]

The variable $VA_IB[ax] determines the inverse-transformed encoder position of an axis in BCS. The BCS value contains all axis-related superimposition portions (DRF, AA_OFF, ext. zero offset etc.) and offset values (CEC, etc.).

Note

VA_ITR, VA_IBC, $VA_IW, $VA_IB

The value of the a variable does not change while reading the variable within an IPO cycle, although the actual value could have changed.

In active transformations, one must consider that the transformation of the actual values into BCS in the IPO cycle can be very time-consuming. In this case one must set an adequate IPO cycle.

7.6 Cartesian PTP travel

Function

This function can be used to approach a Cartesian position with a synchronized axis movement.

It is particularly useful in cases where, for example, the position of the joint is changed, causing the axis to move through a singularity.

When an axis passes through a singularity, the feed velocity would normally be reduced or the axis itself overloaded.

Note

MD24100 $MC_TRAFO_TYPE_1 must be set to the transformation type described in TE4.

The function can only be used meaningfully in conjunction with an active transformation. Furthermore, the "Cartesian PTP travel" function may only be used in conjunction with the G0 and G1 commands. Alarm 14144 "PTP travel not possible" is otherwise output.

When PTP is active, axes in the transformation, e.g. which are traversed using POS, cannot be simultaneously positioning axes. Alarm 17610 is activated to prevent this error.

Activation

The function is activated when the PTP command is programmed.

The function can be deactivated again with the CP command. Both these commands are contained in G group 49.

- PTP command: The programmed Cartesian position is approached with a synchronized axis motion (PTP=point-to-point)
- CP command: The programmed Cartesian point is approached with a path movement (default setting), (CP=continuous path)
- PTPG0 command: The programmed Cartesian PTP motion is performed automatically with each G0 block. The CP command is then set again.

Power On

After `Power on` traversing mode CP is automatically set for axis traversal with transformation. MD20152 $MC_GCODE_RESET_VALUES[48] can be used to switch the default setting to cartesian PTP travel.

Reset

MD20152 $MC_GCODE_RESET_MODE[48] (group 49) defines which setting is active after RESET/end of parts program.

- MD=0: Settings are effected in accordance with machine data

  MD20150 $MC_GCODE_RESET_VALUES[48]

- MD=1: Active setting remains valid

Selection

The setting MD20152 $MC_GCODE_RESET_MODE[48] =0, with MD20150 $MC_GCODE_RESET_VALUES[48] can activate the following:

- MD=2:

  Cartesian PTP travel as previously or

- MD=3:

  PTPG0, traverse only G0 blocks with PTP automatically and then switch over to CP again

Secondary conditions

The following should be noted with respect to tool movement and collision:

- As the PTP command can produce significantly different tool movements to the CP command, any pre-existing subroutines which have been written independently of the active transformation must be adapted to take account of the risks of collision when TRANSMIT is active. This applies particularly in the case of command PTPG0.
- Machine axes always traverse the shortest possible path in response to TRANSMIT and PTP. Minor displacements in the block end point can cause the rotary axis to rotate by -179.99° instead of + 179.99°, even though the block end point has hardly changed.

The following combinations with other NC functions are not legal:

- No tool radius compensation (TRC) may be active with PTP.

  G0 and G41 do not exclude each other in principle. However, an active PTP generates different contours to those computed for the TRC, resulting in the activation of a TRC alarm.

- With PTPG0, for active tool radius compensation (TRC), traverse is by CP.

  Since G0 and G41 do not exclude each other, switch-over to CP is done automatically when tool radius compensation is active. The radius compensation therefore works on the basis of clearly defined contours.

- PTP does not permit smooth approach and retraction (SAR).

  SAR requires a contour in order to construct approach and retraction motion. This information is not available with PTP.

- With PTPG0, CP travel is used for smooth approach and retraction (SAR).

  SAR requires a contour in order to construct approach and retraction motion and to be able to lower and raise tangentially. The blocks required for this purpose are therefore traversed with the CP command. The G0 blocks up to the actual approach contour are executed with PTP and therefore quickly. The same applies to the retract blocks.

- PTP does not permit cutting cycles like CONTPRON, CONTDCON

  Stock removal cycles require a contour to construct the cut segmentation. This information is not available with PTP. Alarm 10931 "Error in cut compensation" is generated in response.
- When PTPG0 is selected, the CP command is applied in cutting cycles such as CONTPRON, CONTDCON. Stock removal cycles require a contour to construct the cut segmentation. The blocks required for this purpose are traversed with the CP command.
- Chamfer and rounding are ignored.
- An axis override in the interpolation must not change during the PTP contour section. This applies, for example, to LIFTFAST, fine tool offset, coupled motion TRAILON and tangential follow-up TANGON.

In PTP blocks

- compressor is automatically deselected because it is not compatible with PTP.
- G643 is automatically switched over to G642.
- Transformation axes must not be configured simultaneously as positioning axes.

Special features

Please take account of the following basic rules with respect to the basic coordinate system:

- Smoothing G642 is always interpreted in the machine coordinate system and not (as usual) in the cartesian basic coordinate system.
- G641 determines the smoothing action as a function of the fictitious path calculated from the machine axis coordinates.
- An F value input with G1 refers to the fictitious path calculated from the machine axis coordinates.

Block search

TRANSMIT during block search can result in different machine axis positions for the same Cartesian position, if a program section is executed with block search.

Interrupts

An illegal action, which may result in a conflict, is rejected with the following alarms:

Alarm 14144: If TRC is selected or activated in PTP. Likewise in PTP with soft approach and retraction (SAR) or PTP without the required G0 and G1 blocks.

Alarm 10753: With PTPG0 and active TRC an internal switch-over to CP is done in order to allow the tool radius correction to be performed correctly.

Alarm 10754: Still possible in case of conflict.

Alarm 10778: Still possible in case of conflict.

Alarm 10744: With PTPG0, CP travel is used for smooth approach and retraction (SAR), in order to ensure correct processing of soft approach and retraction.

Alarm 10746: Still possible in case of conflict.

Alarm 17610: Transformation axes must not be configured simultaneously as positioning axes traversed by means of POS.

Note

For further information about programming plus programming examples, please see:

References:

/PGA/, Programming Manual Job Planning, Chapter Transformations, "Cartesian PTP Travel"

7.6.1 Programming of position

Generally speaking, a machine position is not uniquely defined solely by a position input with Cartesian coordinates and the orientation of the tool. Depending on the kinematics of the relevant machine, the joint may assume up to 8 different positions. These joint positions are specific to individual transformations.

STAT address

A Cartesian position must be convertible into unique axis angles. For this reason, the position of the joints must be entered in the `STAT` address.

The `STAT` address contains a bit for every possible setting as a binary value. The meaning of these bits is determined by the relevant transformation.

As regards the transformations contained in the publication "Handling Transformation Package (TE4)", the bits are assigned to different joint positions, as shown in the figure above.

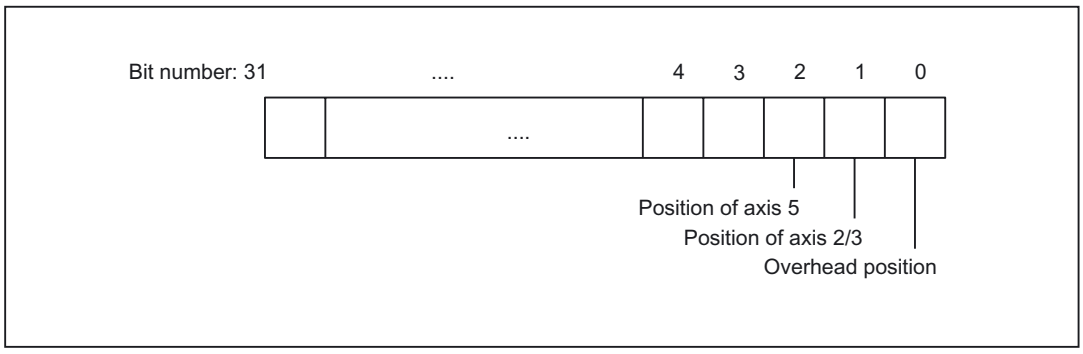


Figure 7-23 Position bits for Handling Transformation Package

Note

It is only meaningful to program the `STAT` address for "Cartesian PTP travel", since changes in position are not normally possible while an axis is traversing with active transformation. The starting point position is applied as the destination point for traversal with the CP command.

7.6.2 Overlap areas of axis angles

TU address

In order to approach axis angles in excess of ±180° without ambiguity, the information must be programmed in the TU (turn) address. The TU address thus represents the sign of the axis angles. This allows an axis angle of |θ| < 360° to be traversed without ambiguity.

Variable TU contains a bit, which indicates the traversing direction for every axis involved in the transformation.

- TU bit=0: 0° ≤ θ < 360°
- TU bit=1: 360° < θ < 0°

The TU bit is set to 0 for linear axes.

In the case of axes with a traversing range >±360°, the axis always moves across the shortest path, because the axis position cannot be specified uniquely by the TU information.

If no TU is programmed for a position, the axis always traverses via the shortest possible route.

7.6.3 Examples of ambiguities of position

The kinematics for a 6axis joint have been used to illustrate the ambiguities caused by different joint positions.

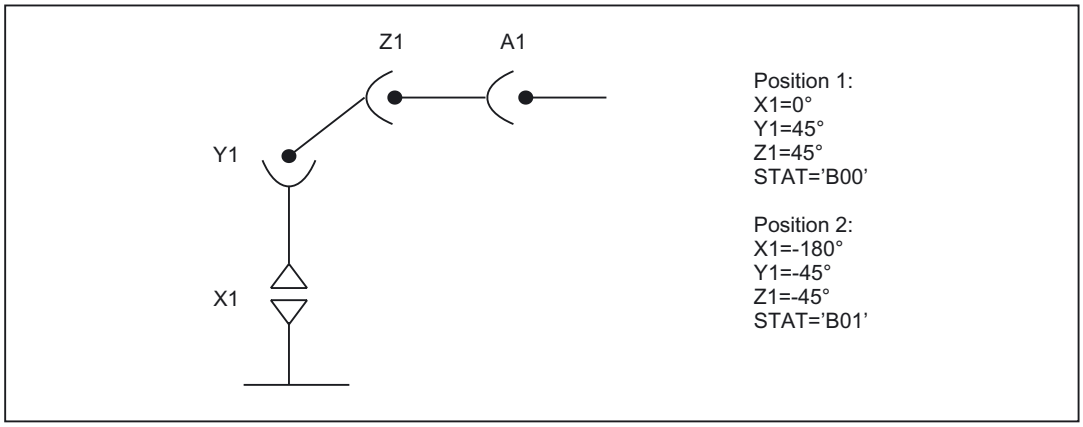


Figure 7-24 Ambiguity in overhead area

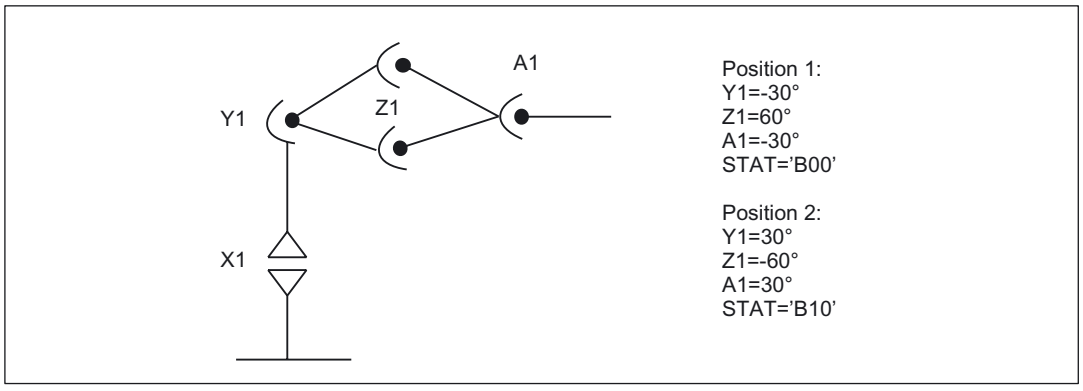


Figure 7-25 Ambiguity of top or bottom elbow

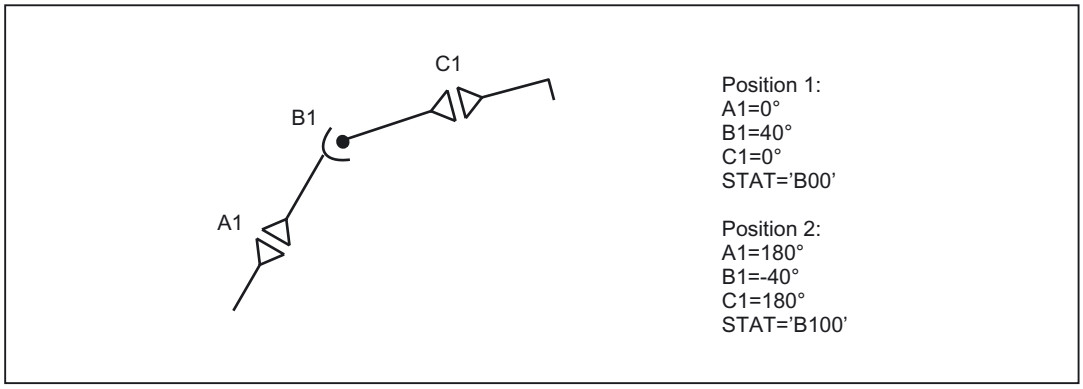


Figure 7-26 Ambiguity of axis B1

7.6.4 Example of ambiguity in rotary axis position

The rotary axis position shown in the following diagram can be approached in the negative or positive direction. The direction is programmed under address A1.

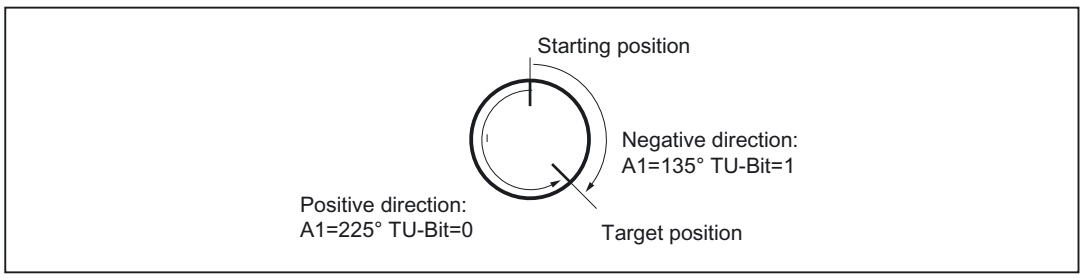


Figure 7-27 Ambiguity in rotary axis position

7.6.5 PTP/CP switchover in JOG mode

In JOG mode, the transformation can be switched on and off via a PLC control signal. This control signal is active only in JOG mode and when a transformation has been activated via the program.

If the mode is switched back to AUTO, the state which was last active before switchover is made active again.

The "point-to-point traversal active" signal DBX317.6 shows which traversal type is active. By means of the "Activate point-to-point traversal" signal DBX29.4 the traversal type can be modified.

Mode change

The "Cartesian PTP travel" function is only useful in the AUTO and MDA modes of operation. The CP setting is automatically activated if the operating mode is switched to JOG. If the mode is then switched back to AUTO or MDA, the mode that was last active in either mode is made active again.

REPOS

The setting for "Cartesian PTP travel" is not altered during re-positioning. If PTP was set in the interruption block, then repositioning takes place in PTP. For an inclined axis "TRAANG", only CP travel is active in `REPOS` mode.

7.7 Cartesian manual travel (optional)

Note

The "Handling transformation package" option is necessary for the "Cartesian manual travel" function.

Function

The "Cartesian manual travel" function, as a reference system for JOG mode, allows axes to be set independently of each other in the following Cartesian coordinate systems:

- Basic coordinate system (BCS)
- Workpiece coordinate system (WCS)
- Tool coordinate system (TCS)

Adjustment and activation is done using machine data:

MD21106 $MC_CART_JOG_SYSTEM (coordinate systems for Cartesian JOG)

Bit	Significance
0	Basic coordinate system
1	Workpiece coordinate system
2	Tool coordinate system

Note

The workpiece coordinate system has been shifted and rotated compared to the basic coordinate system via frames.

Reference:
Function Manual Basic Functions; Axes, Coordinate Systems, Frames (K2)

Representation of the reference system in the coordinate system:

	**WCS** = Workpiece zero		**TCS** = Tool reference point

Selecting reference systems

For JOG motion, one of three reference systems can be specified separately both for

Translation (coarse traverse) with geometry axes, as well as for

Orientation with orientation axes via the

SD42650 $SC_CART_JOG_MODE.

If more than one bit is set for the translation or orientation reference system, or when an attempt is made to set reference system which was not released by the MD21106 $MC_CART_JOG_SYSTEM, the alarm 14148 "Reference system for Cartesian manual travel not allowed" will be generated.

Translation

A translation movement can be used to move the tool tip (TCP) in parallel and 3-dimensional to the axes of the reference system. The traversing movement is made via the VDI signals of the geometry axes.

Machine data MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_x[n] is used to assign the geometry axes. Simultaneous traversing in more than one direction permits the execution of movements that lie parallel to the directions of the reference system.

Translation in the BCS

The basic coordinate system (BCS) describes the Cartesian zero of the machine.

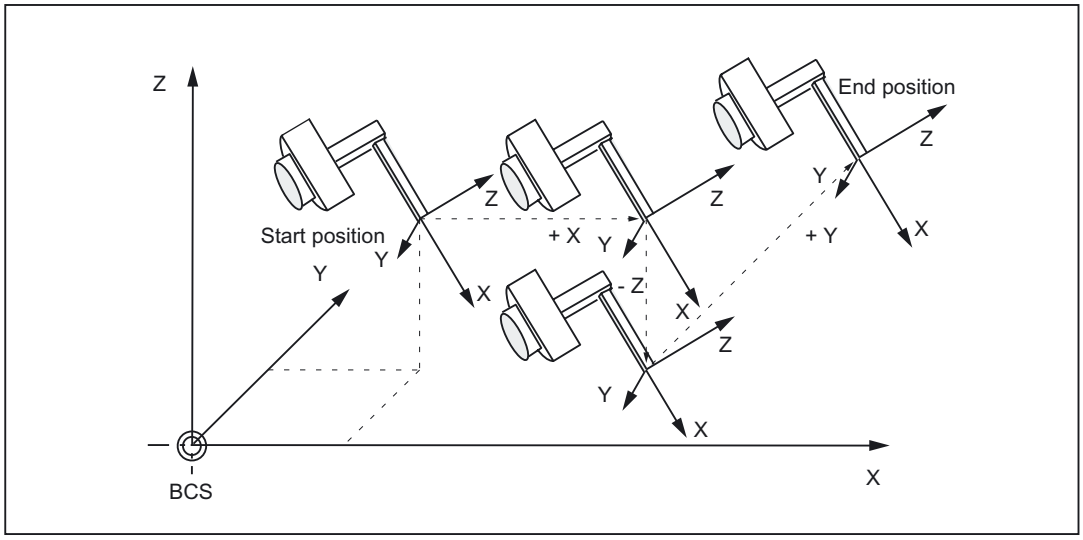


Figure 7-28 Cartesian manual travel in the basic coordinate system (translation)

Translation in the WCS

The workpiece coordinate system (WCS) lies in the workpiece zero. The workpiece coordinate system can be shifted and rotated relative to the reference system via frames. As long as the frame rotation is active, the traversing movements correspond to the translation of the movements in the basic coordinate system.

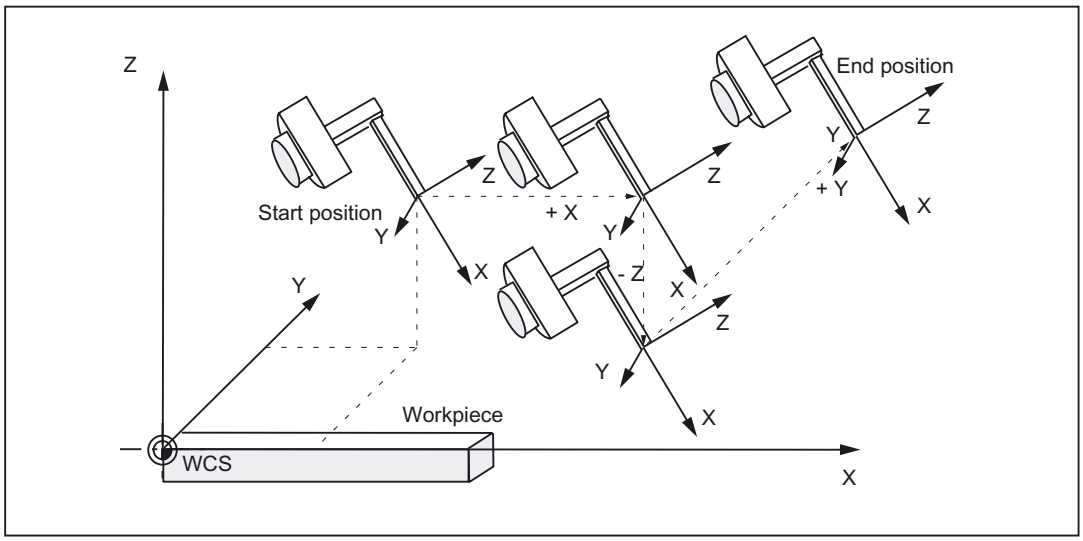


Figure 7-29 Cartesian manual travel in the workpiece coordinate system (translation)

Translation in the TCS

The tool coordinate system (TCS) lies in the tool tip. Its direction depends on the current setting of the machine, since the tool coordinate system moves during the motion.

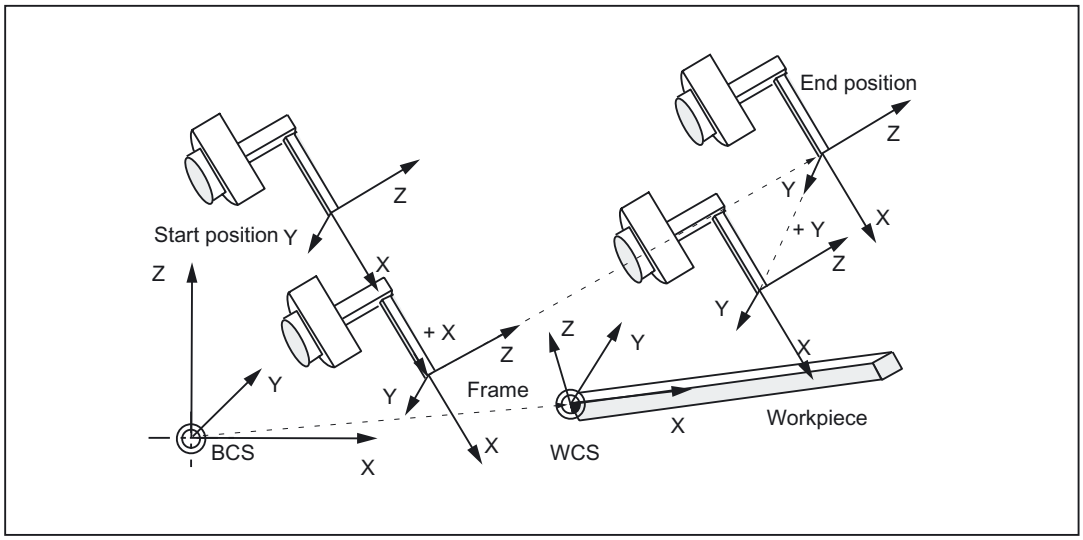


Figure 7-30 Cartesian manual travel in the tool coordinate system (translation)

Translation and orientation in the TCS simultaneously

If translation and orientation motions are executed at the same time, the translation is always traversed corresponding to the current orientation of the tool. This permits infeed movements that are made directly in the tool direction or movements that run perpendicular to tool direction.

Orientation

The tool can be aligned to the component surface via an orientation movement. The orientation movement is given control from the PLC via the VDI signals of the orientation axes (DB21, ... DBB321).

Several orientation axes can be traversed simultaneously. The virtual orientation axes execute rotations around the fixed axes of the relevant reference system.

The **rotations** are identified according to the RPY angles.

- A angle: Rotation around the Z axis
- B angle: Rotation through the Y axis
- C angle: Rotation around the X axis

Programming rotations:

The user can define how rotations are to be executed using the current G codes of group 50 for orientation definition

Specifying `ORIEULER`, `ORIRPY`, `ORIVIRT1` and `ORIVIRT2`.

With ORIVIRT1 rotation is executed according to MD21120 $MC_ORIAX_TURN_TAB_1. The orientation axes are assigned to the channel axes via machine data: MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1.

The **direction of rotation** is determined according to the "right hand rule". The thumb points in the direction of the rotary axis. The finger stipulates the positive direction of rotation.

Orientation in WCS

The rotations are made around the defined directions of the workpiece coordinate system. If frame rotation is active, the movements correspond to the rotations in the basic coordinate system.

Orientation in BCS

The rotations are made around the defined directions of the basic coordinate system.

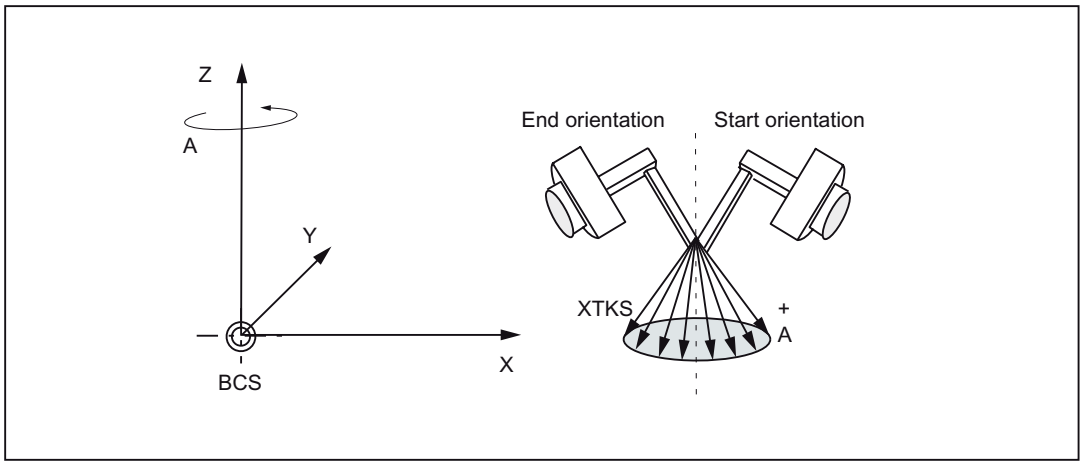


Figure 7-31 Cartesian manual travel in the basic coordinate system orientation angle A

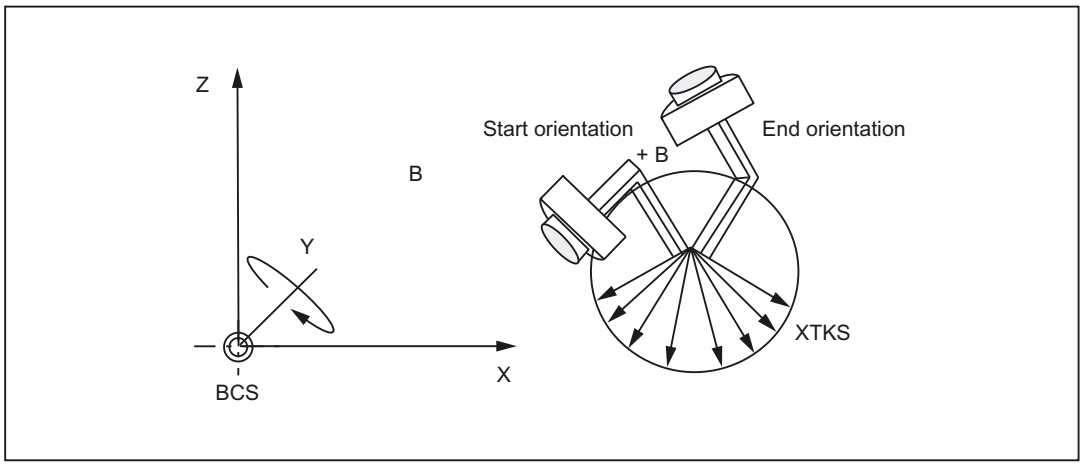


Figure 7-32 Cartesian manual travel in the basic coordinate system orientation angle B

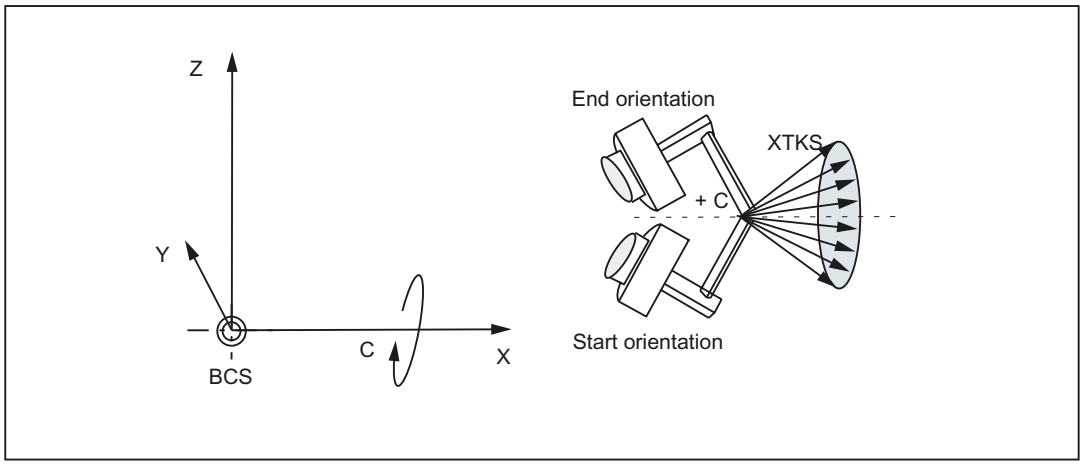


Figure 7-33 Cartesian manual travel in the basic coordinate system orientation angle C

Orientation in TCS

The rotations are around the moving directions in the tool coordinate system. The current homing directions of the tool are always used as rotary axes.

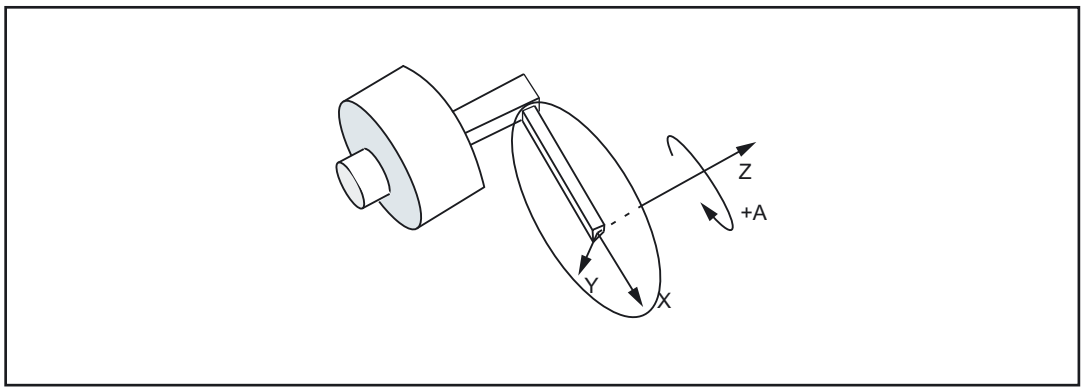


Figure 7-34 Cartesian manual travel in the tool coordinate system, orientation angle A

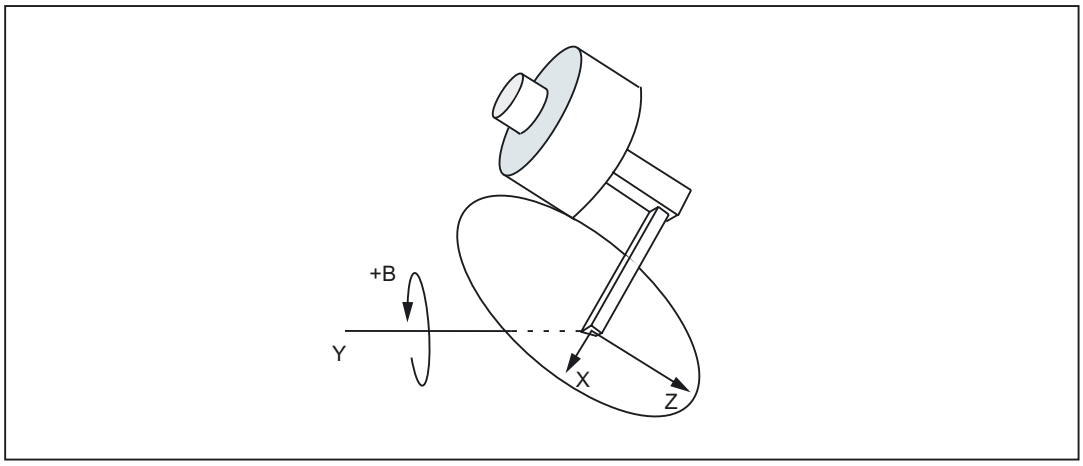


Figure 7-35 Cartesian manual travel in the tool coordinate system, orientation angle B

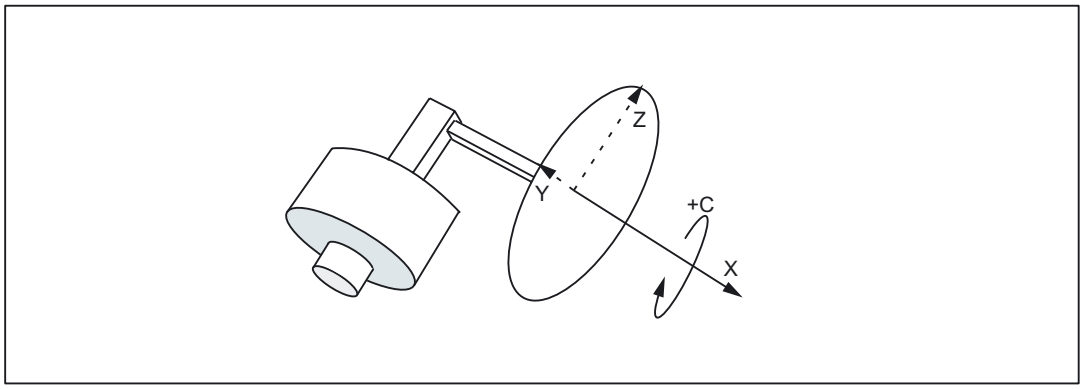


Figure 7-36 Cartesian manual travel in the tool coordinate system, orientation angle C

Secondary conditions

If only NST DB31, ... DBX33.6 ("Transformation active") is on 1, is it possible to execute the Cartesian manual travel function. The following secondary conditions apply:

- SINUMERIK 840D requires the "Handling Transformation Package" option with 5-axis or 6-axis transformation.
- Virtual orientation axes must be defined via the following machine data:

  MD24585 $MC_TRAFO5_ORIAX_ASSIGN_TAB_1[n]
- The NST DB31, ... DBX29.4 ("Activate PTP/CP travel") must be 0.
- Machine data MD21106 $MC_CART_JOG_SYSTEM must be > 0.

Table 7- 2 Conditions for Cartesian manual travel

Transformation in program active (TRAORI..)	G codes PTP/CP	IS "Activate PTP/CP travel"	IS "Transformation active"
FALSE	Not functional	Not functional	DB31, ... DBX33.6 = 0
TRUE	CP	DB31, ... DBX29.4 = 0	DB31, ... DBX33.6 = 1
TRUE	CP	DB31, ... DBX29.4 = 1	DB31, ... DBX33.6 = 0
TRUE	PTP	DB31, ... DBX29.4 = 0	DB31, ... DBX33.6 = 1
TRUE	PTP	DB31, ... DBX29.4 = 1	DB31, ... DBX33.6 = 0

The G code PTP/CP currently active in the program does not affect Cartesian manual travel. The VDI interface signals are interpreted in the channel DB for geometry and orientation axes.

Activation

The reference system for Cartesian manual travel is set as follows:

- The Cartesian manual travel function is activated with the following machine data:

  MD21106 $MC_CART_JOG_SYSTEM > 0

  The BCS, WCS or TCS reference systems are enabled by setting the bits in MD 21106 $MC_CART_JOG_SYSTEM.

- JOG traverse motion via SD42650 SC_CART_JOG_MODE

  Standard behavior as before: Bits 0 to 2 = 0, bits 8 to 10 = 0.

  Reference system for translation via bits 0-2 and the reference system for orientation via bits 8-10.

  If not all of the bits are set to 0, the process uses the new function. The reference systems for translation and orientation may be set independently.

The meaning of the bits is explained in the table below.

Table 7- 3 Bit assignment for SD42650 $SC_CART_JOG_MODE (only one bit may be set)

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved					Translation in the TCS	Translation in the WCS	Translation in the BCS
Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Reserved					Orientation in TCS	Orientation in WCS	Orientation in BCS

Combining reference systems

The table below shows all the combination options for reference systems.

Table 7- 4 Combination options for reference systems

SD42650 $SC_CART_JOG_MODE						Reference system for	
Bit 10	Bit 9	Bit 8	Bit 2	Bit 1	Bit 0	Orientation	Translation
0	0	0	don't care	don't care	don't care	Standard	Standard
Standard	Standard	Standard	0	0	0	Standard	Standard
0	0	1	0	0	1	BCS	BCS
0	0	1	0	1	0	BCS	WCS
0	0	1	1	0	0	BCS	TCS
0	1	0	0	0	1	WCS	BCS
0	1	0	0	1	0	WCS	WCS
0	1	0	1	0	0	WCS	TCS
1	0	0	0	0	1	TCS	BCS
1	0	0	0	1	0	TCS	WCS
1	0	0	1	0	0	TCS	TCS

7.8 Activating transformation machine data via parts program/softkey

7.8.1 Functionality

Transformation MD can now be activated by means of a program command softkey, i.e. these can, for example, be written from the parts program, thus altering the transformation configuration completely.

Up to ten different transformations can be set in the control system. The transformation type is set in the following machine data:

MD24100 $MC_TRAFO_TYPE_1

up to

MD24460 $MC_TRAFO_TYPE_10.

Characteristics

Transformation machine data are NEWCONFIG effective.

The protection level is now 7/7 (KEYSWITCH_0), which means that data can be modified from the NC program without any particular authorization.

Provided that no transformation is selected (activated) when a `NEWCONF` command is issued (regardless whether via the NEWCONF NC program command, the HMI or implicitly following Reset or end of program), the machine data listed above can be altered without restriction and then activated.

Of particular relevance is that new transformations can be configured or existing transformations replaced by one of a different type or deleted, since the modification options are not restricted to re-parameterization of existing transformations.

7.8.2 Constraints

Change machine data

The machine data which affect an active transformation may not be altered; any attempt to do so will generate an alarm.

These are generally all machine data assigned to a transformation via the associated transformation data group. Machine data that are included in the group of an active transformation, but not in use, can be altered (although this would hardly be meaningful). For example, it would be possible to change machine data MD24564 $MC_TRAFO5_NUTATOR_AX_ANGLE_n for an active transformation with MD24100 $MC_$MC_TRAFO_TYPE = 16 (5-axis transformation with rotatable tool and two mutually perpendicular rotary axes A and B) since this particular machine data is not involved in the transformation.

Please note that machine data MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE may not be altered for an active orientation transformation.

Note

In the case of a program interruption (Repos, deletion of distance to go, ASUBs, etc.), the control system requires a number of different blocks that have already been executed for the repositioning operation. The rule forbidding the machine data of an active transformation to be altered also refers to these blocks.

Example:

Two orientation transformations are set via machine data, e.g. MD24100 $MC_TRAFO_TYPE_1 = 16, MD24200 $MC_TRAFO_TYPE_2 = 18.

Assume that the second transformation is active when the NEWCONFIG command is executed. In this case, all machine data that relate only to the first transformation may be changed, e.g.:

MD24500 $MC_TRAFO5_PART_OFFSET_1

but not, for instance:

MD24650 $MC_TRAFO5_BASE_TOOL_2

or

MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE

Furthermore, another transformation (TRANSMIT) can be set, for example with MD24300 $MC_TRAFO_TYPE_3 = 256 and can be parameterized with additional machine data.

Defining geometry axes

Geometry axes must be defined **before** starting the control system with the following machine data:

MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_X[n]

or

MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[n]

Changing the assignment

The assignment of a transformation data set to a transformation is determined by the sequence of entries in MD24100 $MC_TRAFO_TYPE_X. The first entry in the table is assigned to the first transformation data set, and accordingly the second entry to the second data set. This assignment may (and can) not be altered for an active transformation.

Example:

Three transformations are set, two orientation transformations and one Transmit transformation, e.g.

MD24100 $MC_TRAFO_TYPE_1 = 16

; orientation transformation, 1st orientation trafo data set

MD24200 $MC_TRAFO_TYPE_2 = 256: Transmit transformations

MD24300 $MC_TRAFO_TYPE_3 = 18

; orientation transformation, 2nd orientation trafo data set

The first data set for orientation transformations is assigned to the first transformation (equaling the first orientation transformation) and the second transformation data set to the third transformation (equaling the second orientation transformation).

If the third transformation is active when the `NEWCONFIG` command is executed, it is not permissible to change the first transformation into a transformation of another group (e.g. TRACYL) since, in this case, the third transformation would then not become the second orientation transformation, but the first.

In the above example, however, it is permissible to set another orientation transformation for the first transformation (e.g. using MD24100 $MC_TRAFO_TYPE_1 = 32) or a transformation from another group as the first transformation (e.g. using $MD24100 $MC_TRAFO_TYPE_1 = 1024, TRAANG), if the second transformation is changed into an orientation transformation at the same time, e.g. with MD24200 $MC_TRAFO_TYPE_2 = 48.

7.8.3 Control response to power ON, mode change, RESET, block search, REPOS

With the aid of the following machine data it is possible to select a transformation automatically in response to `RESET` (i.e. at end of program as well) and/or on program start:

MD20110 $MC_RESET_MODE_MASK

MD20112 $MC_START_MODE_MASK

and

MD20140 $MC_TRAFO_RESET_VALUE

This may result in the generation of an alarm, for example, at the end or start of a program, if the machine data of an active transformation has been altered.

To avoid this problem when re-configuring transformations via an NC program, we therefore recommend that NC programs are structured as follows:

```
N10 TRAFOOF()                                ; Select a possibly still active
                                               transformation
N20$MC_TRAFO5_BASE_TOOL_1[0]=0               ; Enter machine data
N30$MC_TRAFO5_BASE_TOOL_1[0]=3               ;
N40$MC_TRAFO5_BASE_TOOL_1[0]=200             ;
N130 NEWCONF                                 ; Newly entered machine data
                                             ; take over
N140 M30
```

7.8.4 List of machine data affected

Machine data which can be made NEWCONFIG compatible are listed below.

All transformations

Machine data which are relevant for all transformations:

- MD24100 $MC_TRAFO_TYPE_1 to MD24480 $MC_TRAFO_TYPE_10
- MD24110 $MC_TRAFO_AXES_IN_1 to MD24482 $MC_TRAFO_AXES_IN_10
- MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 to
  MD24484 $MC_TRAFO_GEOAX_ASSIGN_TAB_10

Orientation transformations

Machine data which are relevant for orientation transformations:

- MD24550 $MC_TRAFO5_BASE_TOOL_1 and
  MD24650 $MC_TRAFO5_BASE_TOOL_2
- MD24558 $MC_TRAFO5_JOINT_OFFSET_1 and
  MD24658 $MC_TRAFO5_JOINT_OFFSET_2
- MD24500 $MC_TRAFO5_PART_OFFSET_1 and
  MD24600 $MC_TRAFO5_PART_OFFSET_2
- MD24510 $MC_TRAFO5_ROT_AX_OFFSET_1 and
  MD24610 $MC_TRAFO5_ROT_AX_OFFSET_2
- MD24520 $MC_TRAFO5_ROT_SIGN_IS_PLUS_1 and
  MD24620 $MC_TRAFO5_ROT_SIGN_IS_PLUS_2
- MD 24530: TRAFO5_NON_POLE_LIMIT_1 and
  MD24630 $MC_TRAFO5_NON_POLE_LIMIT_2

- MD24540 $MC_TRAFO5_POLE_LIMIT_1 and
  MD24640 $MC_TRAFO5_POLE_LIMIT_2
- MD24570 $MC_TRAFO5_AXIS1_1 and
  MD24670 $MC_TRAFO5_AXIS1_2
- MD24572 $MC_RAFO5_AXIS2_1 and
  MD24672 $MC_TRAFO5_AXIS2_2
- MD24574 $MC_TRAFO5_BASE_ORIENT_1 and
  MD24674 $MC_TRAFO5_BASE_ORIENT_2
- MD24562 $MC_TRAFO5_TOOL_ROT_AX_OFFSET_1 and
  MD24662 $MC_TRAFO5_TOOL_ROT_AX_OFFSET_2
- MD24564 $MC_TRAFO5_NUTATOR_AX_ANGLE_1 and
  MD24664 $MC_TRAFO5_NUTATOR_AX_ANGLE_2
- MD24566 $MC_TRAFO5_NUTATOR_VIRT_ORIAX_1 and
  MD24666 $MC_TRAFO5_NUTATOR_VIRT_ORIAX_2

Transmit transformations

Machine data which are relevant for Transmit transformations:

- MD24920 $MC_TRANSMIT_BASE_TOOL_1 and
  MD24970 $MC_TRANSMIT_BASE_TOOL_2
- MD24900 $MC_TRANSMIT_ROT_AX_OFFSET_1 and
  MD24950 $MC_TRANSMIT_ROT_AX_OFFSET_2
- MD24910 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_1 and
  MD24960 $MC_TRANSMIT_ROT_SIGN_IS_PLUS_2
- MD24911 MC_RANSMIT_POLE_SIDE_FIX_1 and
  MD24961 $MC_TRANSMIT_POLE_SIDE_FIX_2

Tracyl transformations

Machine data which are relevant for Tracyl transformations:

- MD24820 $MC_TRACYL_BASE_TOOL_1 and
  MD24870 $MC_TRACYL_BASE_TOOL_2
- MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 and
  MD24850 $MC_TRACYL_ROT_AX_OFFSET_2
- MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 and
  MD24870 $MC_TRACYL_ROT_SIGN_IS_PLUS_2
- MD24808 $MC_TRACYL_DEFAULT_MODE_1 and
  MD24858 $MC_TRACYL_DEFAULT_MODE_2

Inclined axis transformations

Machine data which are relevant for inclined axis transformations:

- MD24710 $MC_TRAANG_BASE_TOOL_1 and
  MD24760 $MC_TRAANG_BASE_TOOL_2
- MD24700 $MC_TRAANG_ANGLE_1 and
  MD24750 $MC_TRAANG_ANGLE_2
- MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 and
  MD24770 $MC_TRAANG_PARALLEL_VELO_RES_2
- MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 and
  MD24771 $MC_TRAANG_PARALLEL_ACCEL_RES_2

Chained transformations

Machine data which are relevant for chained transformations:

- MD24995 $MC_TRACON_CHAIN_1 and
  MD24996 $MC_TRACON_CHAIN_2
- MD24997 $MC_TRACON_CHAIN_3 and
  MD24998 $MC_TRACON_CHAIN_4

Persistent transformation

Machine data which are relevant for persistent transformations:

- MD20144 $MC_TRAFO_MODE_MASK
- MD20140 $MC_TRAFO_RESET_VALUE
- MD20110 $MC_RESET_MODE_MASK and
  MD20112 $MC_START_MODE_MASK

Not transformation-specific

Machine data that are not transformation-specific. they are not uniquely assigned to a particular transformation data set or they are relevant even when a transformation is not active:

- MD21110 $MC_X_AXIS_IN_OLD_X_Z_PLANE
- MD21090 $MC_ MAX_LEAD_ANGLE
- MD21092 $MC_ MAX_TILT_ANGLE
- MD21100 $MC_ORIENTATION_IS_EULER

7.9 Constraints

7.9.1 Chained transformations

Two transformations can be chained.

However, not just any transformation can be chained to another one.

In this case, the following restrictions apply:

- The **first** transformation of the chain has to be one of the following transformations:
  - Orientation transformation (3-axis, 4-axis, 5-axis transformations, universal milling head).
  - Transmit
  - Surface line transformation
  - Inclined axis
- The **second** transformation must be an**inclined** axis transformation.
- Only two transformations may be chained.

It is allowed (for test purposes, for instance), to enter only a single transformation into the chaining list.

7.10 Examples

7.10.1 TRANSMIT

The following example relates to the configuration illustrated in the following figure and shows the sequence of main steps required to configure the axes and activate TRANSMIT.

```
; General axis configuration for rotation
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"          : Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"          : Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"          : Geometry axis
MD20060 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1        : X as channel axis 1
MD20060 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0        : Y no channel axis
MD20060 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 2        : Z as channel axis 2
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="ZC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="CC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="ASC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4] = " "
MD20070 $MC_AXCONF_MACHAX_USED[0] = 2             : XC as machine axis 2
```

```
MD20070 $MC_AXCONF_MACHAX_USED[1]=3          : ZC as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[2]=1          : CC as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4        : ASC as machine axis 4
MD20070 $MC_AXCONF_MACHAX_USED[3] = 0        : Empty
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1   : C is spindle 1
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0   : X is no spindle
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0   : Z is no spindle
MD20070 $MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 2   : AS is spindle 2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"   : 1st machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"   : 2nd machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="ZM"   : 3rd machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ASM"  : 4th machine axis
```

```
; prepare for TRANSMIT (as first and only transformation)
$MA_ROT_IS_MODULO[3] = TRUE                  ; c as modulo axis
MD24100 $MC_TRAFO_TYPE_1 = 256               ;TRANSMIT transformation
MD24110 $MC_TRAFO_AXES_IN_1[0] = 1 ;         ; channel axis perpendicular to rotary
                                             axis
MD24110 $MC_TRAFO_AXES_IN_1[1] = 3           ; channel rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[2]=2             ; channel axis parallel to rotary axis
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[         ; 1st channel axis becomes GEOAX X
0]=1
MD24120                                      ; 2nd channel axis becomes GEOAX Y
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=3
MD24120                                      ; 3rd channel axis becomes GEOAX Z
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=2
MD24900                                      ; rotation position X-Y plane against
$MC_TRANSMIT_ROT_AX_OFFSET_1=0               zero position of the rotary axis
MD24910                                      ; Rotary axis turns
$MC_TRANSMIT_ROT_SIGN_IS_PLUS_1=FALSE
MD24920 $MC_TRANSMIT_BASE_TOOL_1             ; tool center distance in X
[0]=0.0
MD24920 $MC_TRANSMIT_BASE_TOOL_1             ; tool center distance in Y
[1]=0.0
MD24920 $MC_TRANSMIT_BASE_TOOL_1             ; tool center distance in Z
[2]=0.0
; activation TRANSMIT
; Programming in X,Y,Z
; Return to rotational operation
TRAFOOF
```

7.10.2 TRACYL

The following figure shows an example relating to the configuration of axes and shows the sequence of main steps required to configure the axes up to activation by TRACYL.

```
; General axis configuration for rotation
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"          ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1]="Y"          ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"          ; Geometry axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 1        ; X as channel axis 1
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 2        ; Y no channel axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 3        ; Z as channel axis 2
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0]="XC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1]="YC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2]="ZC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="CC"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4]="ASC"
MD20070 $MC_AXCONF_MACHAX_USED[0] = 2             ; X as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[1] = 3             ; Y as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[2] = 4             ; Z as machine axis 4
MD20070 $MC_AXCONF_MACHAX_USED[3] = 1             ; C as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[4] = 5             ; AS as machine axis 5
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1        ; C is spindle 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0        ; X is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0        ; Y is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 0        ; Z is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX5]= 2        ; AS is spindle 2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]="CM"        ; 1st machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]="XM"        ; 2nd machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]="YM"        ; 3rd machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="ZM"        ; 4th machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[4]="ASM"       ; 5th machine axis
```

```
; prepare for TRACYL (as first and only transformation)
MD24100 $MC_TRAFO_TYPE_1 = 513 ;      ; Transformation TRACYL with groove wall
                                        offset
MD24110 $MC_TRAFO_AXES_IN_1[0] = 1    ; channel axis radial to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[1] = 4    ; channel axis in generated cylinder surface
                                        perpendicular to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3    ; channel axis parallel to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[3] = 2    ; channel axis special axis to index [0]
```

```
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1     ; 1st channel axis becomes GEOAX X
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 4     ; 2nd channel axis becomes GEOAX Y
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]= 3       ; 3rd channel axis becomes GEOAX Z
MD24800 $MC_TRACYL_ROT_AX_OFFSET_1 = 0           ; rotation position X-Y plane against zero position of the rotary axis
MD24810 $MC_TRACYL_ROT_SIGN_IS_PLUS_1 = FALSE    ; rotary axes turns
MD24820 $MC_TRACYL_BASE_TOOL_1 [0] = 0.0         ; tool center distance in X
MD24820 $MC_TRACYL_BASE_TOOL_1 [1] = 0.0         ; tool center distance in Y
MD24820 $MC_TRACYL_BASE_TOOL_1 [2] = 0.0         ; tool center distance in Z
; activation TRACYL(40.0)
; programming in Y and Z see below
; Return to rotational operation
TRAFOOF
```

Programming with groove wall offset

(TRAFO_TYPE_n=513)

Contour

It is possible to produce a groove which is wider than the tool by using address OFFN to program the compensation direction (G41, G42) in relation to the programmed reference contour and the distance of the groove side wall from the reference contour (see fig.).

Tool radius

The tool radius is automatically taken into account with respect to the groove side wall (see figure). The full functionality of the plane tool radius compensation is available (steady transition at outer and inner corners as well as solution of bottleneck problems).

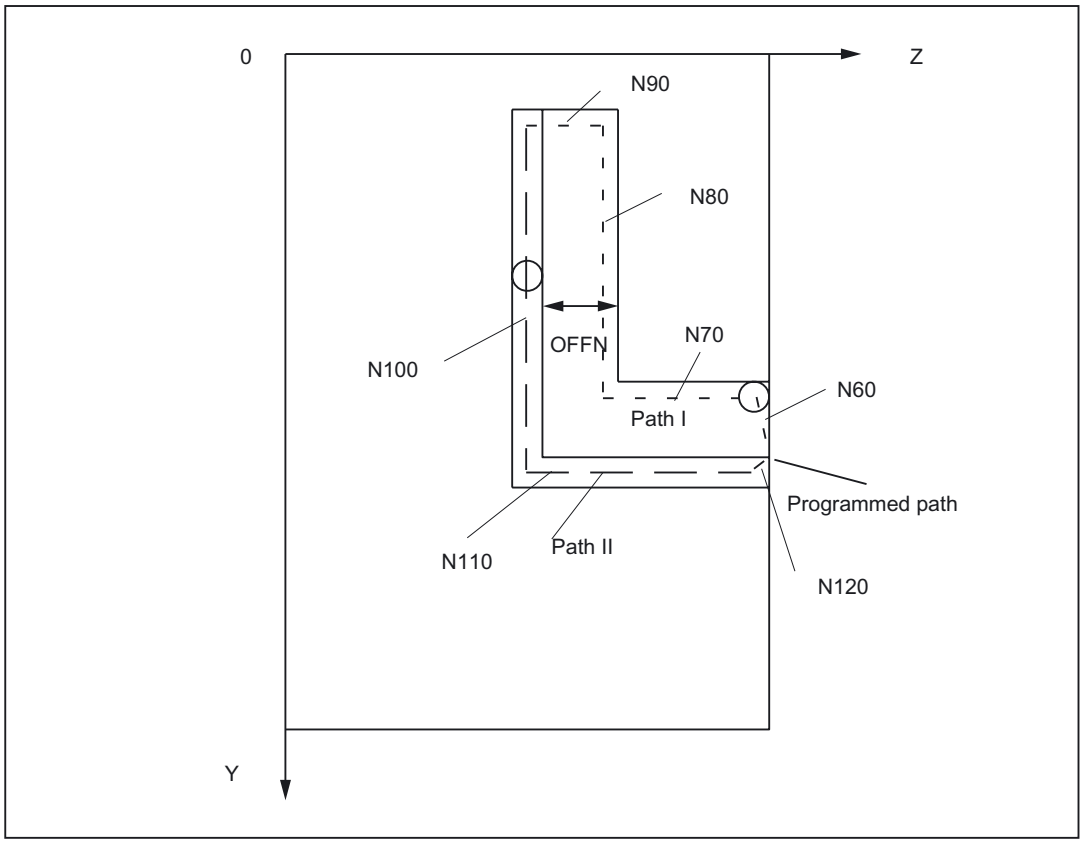


Figure 7-37 Groove with wall compensation, cylinder coordinates (simplified sketch)

; Example program, which guides the tool after transformation selection

; on path I via path II back to the starting position

; (machine data see "Data Description", Example X-Y-Z-C kinematics):

```
N1 SPOS=0;                                ; Take-over of spindle into rotary axis
                                          operation
N5 G0 X25 Y0 Z105 CC=200 F5000 G64        ; Positioning of machine above groove
                                          center
N10 TRACYL(40.)                           ;Transformation selection with reference
                                          diameter
                                          :40 mm
N20 G19 G90                               ; Operating plane is cylinder surface
N30 T1 D1                                 ; Tool selection, can also be before
                                          TRACY (..)
N40 G1 X20                                ;Infeed tool to groove base
```

```
N50 OFFN=12                                    ; Determine groove wall distance, need
                                               not be in
                                               ; its own line


; Approach of groove wall
N60 G1 Z100 G42                                ; TRC selection to approach
                                               groove wall
Machining groove sector path I
N70 G1 Z50                                     ; Groove part parallel to
                                               cylinder plane
N80 G1 Y10                                     ; Groove part parallel to
                                               circumference
; Approaching groove wall for path II
N90 OFFN=4 G42                                 ; Specifying groove wall
                                               distance and TRC selection
                                               ; for approach of groove wall
; Machining groove sector path II
N100 G1 Y70                                    ; according to CC=200 degrees
N110 G1 Z100                                   ; revert to initial value
;Retract from groove wall
N120 G1 Z105 G40                               ; TRC deselection to retract
                                               from groove wall
N130 G0 X25                                    ; Retract from groove
N140 TRAFOOF N150 G0 X25 Y0 Z105 CC=200 D0     ; go back to starting point
                                               and deselection
                                               ; of the tool offset
```

Programming without groove wall offset

TRACYL without groove wall offset with supplementary linear axis (TRAFO_TYPE_n = 514)

```
; For the following parts program the following machine data settings are
a prerequisite:
MD20070 $MC_AXCONF_MACHAX_USED[0]=1              ; X as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[1] = 2            ; Y as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[2] = 3            ; Z as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4            ; C as machine axis 4
MD20070 $MC_AXCONF_CHANAX_NAME_TAB[1] = "Y2"
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1;    ; X as channel axis 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 2;    ; Y no channel axis
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3;    ; Z as channel axis 2
MD24100 $MC_TRAFO_TYPE_1 = 514                   ; TRACYL without groove
                                                 wall offset only with
                                                 tool length offset
MD24110 $MC_TRAFO_AXES_IN_1[0] = 1               ; channel axis radial to
                                                 rotary axis
```

```
MD24110 $MC_TRAFO_AXES_IN_1[1] = 4              ; Channel axis in
                                                cylinder surface
                                                ; perpendicular to rotary
                                                axis
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3              ; channel axis parallel
                                                to rotary axis
MD24110 $MC_TRAFO_AXES_IN_1[3] = 2              ; Channel axis special
                                                axis to index [0]
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 1    ; 1st channel axis
                                                becomes GEOAX X
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 4    ; 2nd channel axis
                                                becomes GEOAX Y
MD24110 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3    ; 3rd channel axis
                                                becomes GEOAX Z
MD24808 $MC_TRACYL DEFAULT MODE_1 =0            ; or not set at all
```

Tool data:

```
$TC_DP1[1,1] = 120                  ; Tool type shaft miller
$TC_DP2[1,1] = 0
$TC_DP3[1,1] = 0                    ; Length offset vector
$TC_DP4[1,1] = 25
$TC_DP5[1,1] =5
$TC_DP6[1,1] =4                     ; Radius, tool radius
```

Part program:

```
N1001 T1 D1 G54 G19 G90 F5000 G64   ;Selection of the 1st TRACYL without
N1005 G0 X25 Y0 Z105 A=200          groove wall offset
N1010 TRACYL(40.)                   ; Transformation selection
N1040 G1 X20
N1060 G1 Z100
N1070 G1 Z50
N1080 G1 Y10
N1140 TROFOOF
N1150 G0 X25 Y0 Z105 A=200          ; Selection of the 1st TRACYL with
                                    groove wall offset
N2010 G0 TRACYL(40.,1,1)            ; TRACYL (40., ,1) would also be
                                    possible
N2040 G1 X20
N2060 G1 Z100
N2070 G1 Z50
N2080 G1 Y10
N2140 TROFOOF
```

7.10.3 TRAANG

For the configuration shown in Figure "Groove with Groove Wall Offset, Cylinder Coordinates", an example relating to the configuration of axes which shows the sequence of main steps required to configure the axes up to activation by TRAANG is shown.

```
; General axis configuration for grinding
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[0]="X"          ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[1] = " "        ; Geometry axis
MD20060 $MC_AXCONF_GEOAX_NAME_TAB[2]="Z"          ; Geometry axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[0] = 0        ; X no channel axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[1] = 0        ; Y no channel axis
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB[2] = 1        ; Z as channel axis 1
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[0] = "Z"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[1] = "C"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[2] = "AS1"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3] = "MU"
MD20070 $MC_AXCONF_MACHAX_USED[0] = 3             ; Z as machine axis 3
MD20070 $MC_AXCONF_MACHAX_USED[1]=1               ; C as machine axis 1
MD20070 $MC_AXCONF_MACHAX_USED[2] = 4             ; AS as machine axis 4
MD20070 $MC_AXCONF_MACHAX_USED[3] = 2             ; MU as machine axis 2
MD20070 $MC_AXCONF_MACHAX_USED[3] = 0             ; empty
MD20070 $MC_AXCONF_MACHAX_USED[3] = 0             ; empty
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX1]= 1        ; C is spindle 1
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX2]= 0        ; X is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX3]= 0        ; Z is no spindle
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX4]= 2        ; AS is spindle 2
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[0]= "C1"       ; 1st machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[1]= "MU"       ; 2nd machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[2]= "MZ"       ; 3rd machine axis
MD10000 $MN_AXCONF_MACHAX_NAME_TAB[3]="AS1"       ; 4th machine axis
```

```
; prepare for TRAANG (as first and only transformation)
MD24100 $MC_TRAFO_TYPE_1 = 1024                   ; Transformation TRAANG
MD24110 $MC_TRAFO_AXES_IN_1[0]=4                  ; channel axis inclined axis
MD24110 $MC_TRAFO_AXES_IN_1[1]=1                  ; channel axis parallel to axis Z
MD24110 $MC_TRAFO_AXES_IN_1[2] = 0                ; channel axis not active
MD24120
$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=4                 ; X 1st channel axis
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1]
= 0                                               ; Y 2nd channel axis
```

```
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2]    ; Z 3rd channel axis
= 1
MD24700 $MC_TRAANG_ANGLE_1 = 30.            ; Angle of inclined axis
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0      ; tool center distance in X
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0      ; tool center distance in Y
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0      ; tool center distance in Z
TRAANG                                      ; activation
                                            ; Programming in X,Y,Z
TRAFOOF                                     ; Return to rotational operation
```

7.10.4 Chained transformations

Examples

The following chapter determines:

- The general channel configuration
- Single transformations
- Chained transformations consisting of previously defined single transformations
- Activation of single transformations
- Activation of chained transformations

The examples include the following transformations:

- 5-axis transformation with rotatable tool and axis sequence AB (trafo type 16)
- Transmit (trafo type 256)
- Inclined axis (trafo type 1024)
- Chaining of the 1st and 3rd transformation (trafo type 8192)
- Chaining of the 2nd and 3rd transformation (trafo type 8192)

General channel configuration

```
CHANDATA(1)                                 ; Channel data in channel 1
MD20070 $MC_AXCONF_MACHAX_USED[0]=1
MD20070 $MC_AXCONF_MACHAX_USED[1] = 2
MD20070 $MC_AXCONF_MACHAX_USED[2] = 3
MD20070 $MC_AXCONF_MACHAX_USED[3] = 4
MD20070 $MC_AXCONF_MACHAX_USED[4]=5
MD20070 $MC_AXCONF_MACHAX_USED[5]=6
MD20070 $MC_AXCONF_MACHAX_USED[6]=7
MD20070 $MC_AXCONF_MACHAX_USED[7] = 0
```

```
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[3]="A"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[4]="B"
MD20080 $MC_AXCONF_CHANAX_NAME_TAB[5] = "C"
MD36902 $MA_IS_ROT_AX[ AX4 ] = TRUE
MD36902 $MA_IS_ROT_AX[ AX5 ] = TRUE
MD36902 $MA_IS_ROT_AX[ AX6 ] = TRUE
MD36902 $MA_IS_ROT_AX[ AX7 ] = TRUE
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX5]= 0
MD35000 $MA_SPIND_ASSIGN_TO_MACHAX[AX7] = 1
MD35000 $MA_ROT_IS_MODULO[AX7] = TRUE
```

Single transformations

```
; 1. TRAORI
MD24470 $MC_TRAFO_TYPE_1= 16            ; TRAORI: A-B kinematics
MD24410 $MC_TRAFO_AXES_IN_1[0]=1
MD24410 $MC_TRAFO_AXES_IN_1[1]=2
MD24410 $MC_TRAFO_AXES_IN_1[2]=3
MD24410 $MC_TRAFO_AXES_IN_1[3]=4
MD24410 $MC_TRAFO_AXES_IN_1[4]=5
MD24410 $MC_TRAFO_AXES_IN_1[5]=0
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[0]=1
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[1]=2
MD24120$MC_TRAFO_GEOAX_ASSIGN_TAB_1[2]=3
MD24550$MC_TRAFO5_BASE_TOOL_1[0]=0
MD24550$MC_TRAFO5_BASE_TOOL_1[1]=0
MD24550$MC_TRAFO5_BASE_TOOL_1[2]=0


; 2. TRANSMIT
MD24200 $MC_TRAFO_TYPE_2 = 256          ; TRANSMIT
MD24210 $MC_TRAFO_AXES_IN_2[0] = 1
MD24210 $MC_TRAFO_AXES_IN_2[1] = 6
MD24210 $MC_TRAFO_AXES_IN_2[2]=3
MD24210 $MC_TRAFO_AXES_IN_2[3] = 0
MD24210 $MC_TRAFO_AXES_IN_2[4] = 0
MD24210 $MC_TRAFO_AXES_IN_2[5] = 0
MD24210 $MC_TRAFO_AXES_IN_2[6]=0
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =6
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3
```

```
; 3. TRAANG
MD24300 $MC_TRAFO_TYPE_3 = 1024        ; TRAANG
MD24310 $MC_TRAFO_AXES_IN_3[0] = 1
MD24310 $MC_TRAFO_AXES_IN_3[1] = 3
MD24310 $MC_TRAFO_AXES_IN_3[2] = 2
MD24310 $MC_TRAFO_AXES_IN_3[3] = 0
MD24310 $MC_TRAFO_AXES_IN_3[4] = 0
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =1
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =3
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =2
MD24700 $MC_TRAANG_ANGLE_1 = 45.
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0
```

Chained transformations

```
; 4. TRACON (Chaining TRAORI/TRAANG)
MD24400 $MC_TRAFO_TYPE_4 = 8192
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =2
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
MD24995 $MC_TRACON_CHAIN_1[0] = 1
MD24995 $MC_TRACON_CHAIN_1[1] = 3
MD24995 $MC_TRACON_CHAIN_1[2] = 0
; 5. TRACON (Chaining TRANSMIT/TRAANG)
MD24430 $MC_TRAFO_TYPE_5 = 8192
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =6
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3
MD24996 $MC_TRACON_CHAIN_2[0] = 2
MD24996 $MC_TRACON_CHAIN_2[1] = 3
MD24996 $MC_TRACON_CHAIN_2[2] = 0
```

Parts program (extracts)

Example of an NC program which uses the set transformations:

```
; Call single transformations
                       ; Tool specification
$TC_DP1[1,1] = 120     ; Tool type
$TC_DP3[1,1] = 10      ; Tool length
n2 x0 y0 z0 a0 b0 f20000 t1 d1n4 x20
n30 TRANSMIT           ; Switch on TRANSMIT
n40 x0 y20
n50 x-20 y0
n60 x0 y-20
n70 x20 y0
n80 TRAFOOF            ; Switch off TRANSMIT
n130 TRACYL(45.)       ; Activate inclined axis transformation, parameter: Angle 45°
n140 x0 y0 z20
n150 x-20 z0
n160 x0 z-20
n170 x20 z0
```

Note

The above examples assume that the angle of the "inclined axis" can be set on the machine and is set to 0° when the single transformation is activated.

```
; Activate 1st chained transformations
; TRAORI + TRAANG
n230 TRACON(1, 45.)      ; 1st of the 2 chained transformations to be switched on
                         ; The previously active transformation TRAANG is
                         automatically deselected
                         ; The parameter for the inclined axis is 45°
n240 x10 y0 z0 a3=-1 C3 =1 oriwks
n250 x10 y20 b3 = 1 c3 = 1
```

```
; Activate 2nd chained transformations
; TRANSMIT + TRAANG
n330 TRACON(2, 40.)      ; Activate 2nd chained transformation
                         ; The parameter for the inclined axis is 40°
n335 x20 y0 z0
```

```
n340 x0 y20 z10
n350 x-20 y0 z0
n360 x0 y-20 z0
n370 x20 y0 z0
n380 TRAFOOF            ; Deactivate 2nd chained transformation
n1000 M30
```

7.10.5 Activating transformation MD via a parts program

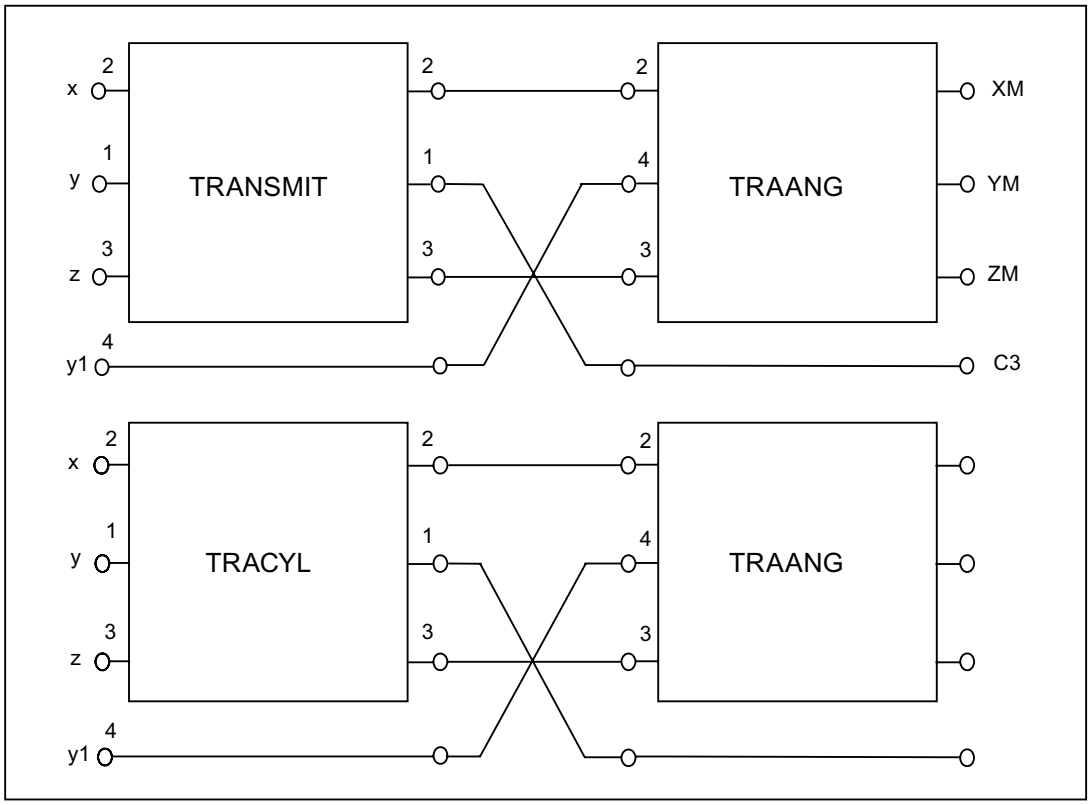
It would be permissible in the following example to reconfigure (write) a machine data affecting the second transformation (e.g. MD24650 $MC_TRAFO5_BASE_TOOL_2[2]) in block N90, since writing a machine data alone does not activate it. However, if the program remained otherwise unchanged, an alarm would occur in block N130, because an attempt would then be made to modify an active transformation.

Example program:

```
N40 TRAORI(2)                              ; Select 2nd orientation transformation
N50 X0 Y0 Z0 F20000 T1 T1
N60 A50 B50
N70 A0 B0
N80 X10
N90 $MC_TRAFO5_BASE_TOOL_1[2] = 50         ; Overwriting machine data of
                                           ; 1st orientation transformation
N100 A20
N110 X20
N120 X0
N130 NEWCONF                               ; Take over new machine data
N140 TRAORI(1)                             ; Select 1st orientation transformation MD
                                           ; becomes effective
N150 G19 X0 Y0 Z0
N160 A50 B50
N170 A0 B0
N180 TRAFOOF
N190 M30
```

7.10.6 Axis positions in the transformation chain

Two chained transformations are configured in the following example, and the system variables for determining the axis positions in the synchronous action are read cyclically in the part program.



Machine data

```
CHANDATA(1)

MD24100 $MC_TRAFO_TYPE_1=256                          ; TRANSMIT
MD24110 $MC_TRAFO_AXES_IN_1[0] = 2
MD24110 $MC_TRAFO_AXES_IN_1[1] = 1
MD24110 $MC_TRAFO_AXES_IN_1[2] = 3
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [0] = 2
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [1] = 1
MD24120 $MC_TRAFO_GEOAX_ASSIGN_TAB_1 [2] = 3

MD24200 $MC_TRAFO_TYPE_2=512                          ; TRACYL
MD24210 $MC_TRAFO_AXES_IN_2[0]=2
MD24210 $MC_TRAFO_AXES_IN_2[1]=1
MD24210 $MC_TRAFO_AXES_IN_2[2]=3
```

```
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[0] =2
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[1] =1
MD24220 $MC_TRAFO_GEOAX_ASSIGN_TAB_2[2] =3

MD24300 $MC_TRAFO_TYPE_3=1024                    ; TRAANG
MD24310 $MC_TRAFO_AXES_IN_3[0] = 2
MD24310 $MC_TRAFO_AXES_IN_3[1]=4
MD24310 $MC_TRAFO_AXES_IN_3[2] = 3
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[0] =2
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[1] =4
MD24320 $MC_TRAFO_GEOAX_ASSIGN_TAB_3[2] =3
MD24700 $MC_TRAANG_ANGLE_1 = 45.
MD24720 $MC_TRAANG_PARALLEL_VELO_RES_1 = 0.2
MD24721 $MC_TRAANG_PARALLEL_ACCEL_RES_1 = 0.2
MD24710 $MC_TRAANG_BASE_TOOL_1 [0] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [1] = 0.0
MD24710 $MC_TRAANG_BASE_TOOL_1 [2] = 0.0
```

1st TRANSMIT/TRAANG chaining

```
MD24400 $MC_TRAFO_TYPE_4=8192                    ; TRACON (1)
MD24995 $MC_TRACON_CHAIN_1[0] = 1
MD24995 $MC_TRACON_CHAIN_1[1] = 3
MD24995 $MC_TRACON_CHAIN_1[2] = 0
MD24995 $MC_TRACON_CHAIN_1[3] = 0
MD24410 $MC_TRAFO_AXES_IN_4[0]=1
MD24410 $MC_TRAFO_AXES_IN_4[1]=2
MD24410 $MC_TRAFO_AXES_IN_4[2]=3
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[0] =2
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[1] =1
MD24420 $MC_TRAFO_GEOAX_ASSIGN_TAB_4[2] =3
```

2nd TRACYL/TRAANG chaining

```
MD24430 $MC_TRAFO_TYPE_5=8192                    ; TRACON (2)
MD24996 $MC_TRACON_CHAIN_2[0] = 2
MD24996 $MC_TRACON_CHAIN_2[1] = 3
MD24996 $MC_TRACON_CHAIN_2[2]=0
MD24996 $MC_TRACON_CHAIN_2[3]=0
MD24432 $MC_TRAFO_AXES_IN_5[0]=1
MD24432 $MC_TRAFO_AXES_IN_5[1]=2
MD24432 $MC_TRAFO_AXES_IN_5[2]=3
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[0] =2
```

MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[1] =1
MD24434 $MC_TRAFO_GEOAX_ASSIGN_TAB_5[2] =3

M17

Part program

```
Program code                                Comment
N10 $TC_DP1[1,1]=120
N20 $TC_DP3[1,1]= 20
N30 $TC_DP4[1,1]=0
N40 $TC_DP5[1,1]=0
N50
N60 X0 Y0 Z0 F20000 T1 D1
N70
N80                                         ; cyclical reading of the variables in the synchronous action
N90 ID=1 WHENEVER TRUE DO $R0=$AA_ITR[X,0] $R1=$AA_ITR[X,1] $R2=$AA_ITR[X,2]
N100 ID=2 WHENEVER TRUE DO $R3=$AA_IBC[X] $R4=$AA_IBC[Y] $R5=$AA_IBC[Z]
N110 ID=3 WHENEVER TRUE DO $R6=$VA_IW[X]-$AA_IW[X]
N120 ID=4 WHENEVER TRUE DO $R7=$VA_IB[X]-$AA_IB[X]
N130 ID=5 WHENEVER TRUE DO $R8=$VA_IBC[X]-$AA_IBC[X]
N140 ID=6 WHENEVER TRUE DO $R9=$VA_ITR[X,1]-$AA_ITR[X,1]
N150
N160                                        ; 1. TRANSMIT/TRAANG chaining
N170 TRACON(1,)
N180 X20 Y0 Z0
N190 X0 Y20 Z10
N200 X-20 Y0 Z0
N210 X0 Y-20 Z0
N220 X20 Y0 Z0
N230 TRAFOOF
N240
N250                                        ; 2. TRACYL/TRAANG chaining
N260 TRACON (2, 40.)
N270 X20 Y0 Z0
N280 X0 Y20 Z10
N290 X-20 Y0 Z0
N300 X0 Y-20 Z0
N310 X20 Y0 Z0
N320 TRAFOOF
N330
N340 M30
```

7.11 Data lists

7.11.1 Machine data

7.11.1.1 TRANSMIT

Channel-specific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
22534	TRAFO_CHANGE_M_CODE	M code for transformation changeover
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Geo-axis assignment for 6th transformation
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24900	TRANSMIT_ROT_AX_OFFSET_1	Deviation of rotary axis from zero position in degrees (1st TRANSMIT)

Number	Identifier: $MC_	Description
24910	TRANSMIT_ROT_SIGN_IS_PLUS_1	Sign of rotary axis for TRANSMIT (1st TRANSMIT)
24911	TRANSMIT_POLE_SIDE_FIX_1	Limitation of working range in front of/behind pole, 1st transformation
24920	TRANSMIT_BASE_TOOL_1	Distance of tool zero point from origin of geo-axes (1st TRANSMIT)
24950	TRANSMIT_ROT_AX_OFFSET_2	Deviation of rotary axis from zero position in degrees (2nd TRANSMIT)
24960	TRANSMIT_ROT_SIGN_IS_PLUS_2	Sign of rotary axis for TRANSMIT (2nd TRANSMIT)
24961	TRANSMIT_POLE_SIDE_FIX_2	Limitation of working range in front of/behind pole, 2nd transformation
24970	TRANSMIT_BASE_TOOL_2	Distance of tool zero point from origin of geo-axes (2nd TRANSMIT)

7.11.1.2 TRACYL

Channel-specific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
20144	TRAFO_MODE_MASK	Selection of the kinematic transformation function
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24130	TRAFO_INCLUDES_TOOL_1	Tool handling with active transformation 1.
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24230	TRAFO_INCLUDES_TOOL_2	Tool handling with active transformation 2.
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24330	TRAFO_INCLUDES_TOOL_3	Tool handling with active transformation 3.
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24426	TRAFO_INCLUDES_TOOL_4	Tool handling with active transformation 4.
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation

Number	Identifier: $MC_	Description
24436	TRAFO_INCLUDES_TOOL_5	Tool handling with active transformation 5.
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Assignment geometry axes for 6th transformation
24446	TRAFO_INCLUDES_TOOL_6	Tool handling with active transformation 6.
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24456	TRAFO_INCLUDES_TOOL_7	Tool handling with active transformation 7.
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24466	TRAFO_INCLUDES_TOOL_8	Tool handling with active transformation 8.
24470	TRAFO_TYPE_9	Definition of the 9th transformation in channel
24472	TRAFO_AXES_IN_9	Axis assignment for the 9th transformation
24474	TRAFO_GEOAX_ASSIGN_TAB_9	Geo-axis assignment for 9th transformation
24476	TRAFO_INCLUDES_TOOL_9	Tool handling with active transformation 9.
24480	TRAFO_TYPE_10	Definition of the 10th transformation in channel
24482	TRAFO_AXES_IN_10	Axis assignment for the 10th transformation
24484	TRAFO_GEOAX_ASSIGN_TAB_10	Geo-axis assignment for 10th transformation
24486	TRAFO_INCLUDES_TOOL_10	Tool handling with active transformation 10.
24800	TRACYL_ROT_AX_OFFSET_1	Deviation of rotary axis from zero position in degrees (1st TRACYL)
24808	TRACYL_DEFAULT_MODE_1	Selection of TRACYL mode (1st TRACYL)
24810	TRACYL_ROT_SIGN_IS_PLUS_1	Sign of rotary axis for TRACYL (1st TRACYL)
24820	TRACYL_BASE_TOOL_1	Distance of tool zero point from origin of geo-axes (1st TRACYL)
24850	TRACYL_ROT_AX_OFFSET_2	Deviation of rotary axis from zero position in degrees (2nd TRACYL)
24858	TRACYL_DEFAULT_MODE_2	Selection of TRACYL mode (2nd TRACYL)
24860	TRACYL_ROT_SIGN_IS_PLUS_2	Sign of rotary axis for TRACYL (2nd TRACYL)
24870	TRACYL_BASE_TOOL_2	Distance of tool zero point from origin of geo-axes (2nd TRACYL)
22534	TRAFO_CHANGE_M_CODE	M code for transformation changeover

7.11.1.3 TRAANG

Channel-specific machine data

Number	Identifier: $MC_	Description
20110	RESET_MODE_MASK	Definition of control basic setting after run-up and RESET/part program end
20140	TRAFO_RESET_VALUE	Basic transformation position
20144	RAFO_MODE_MASK	Selection of the kinematic transformation function
20534	TRAFO_CHANGE_M_CODE	M code for transformation changeover
24100	TRAFO_TYPE_1	Definition of the 1st transformation in channel
24110	TRAFO_AXES_IN_1	Axis assignment for the 1st transformation
24120	TRAFO_GEOAX_ASSIGN_TAB_1	Geo-axis assignment for 1st transformation
24200	TRAFO_TYPE_2	Definition of the 2nd transformation in channel
24210	TRAFO_AXES_IN_2	Axis assignment for the 2nd transformation
24220	TRAFO_GEOAX_ASSIGN_TAB_2	Geo-axis assignment for 2nd transformation
24300	TRAFO_TYPE_3	Definition of the 3rd transformation in channel
24310	TRAFO_AXES_IN_3	Axis assignment for the 3rd transformation
24320	TRAFO_GEOAX_ASSIGN_TAB_3	Geo-axis assignment for 3rd transformation
24400	TRAFO_TYPE_4	Definition of the 4th transformation in channel
24410	TRAFO_AXES_IN_4	Axis assignment for the 4th transformation
24420	TRAFO_GEOAX_ASSIGN_TAB_4	Geo-axis assignment for 4th transformation
24430	TRAFO_TYPE_5	Definition of the 5th transformation in channel
24432	TRAFO_AXES_IN_5	Axis assignment for the 5th transformation
24434	TRAFO_GEOAX_ASSIGN_TAB_5	Geo-axis assignment for 5th transformation
24440	TRAFO_TYPE_6	Definition of the 6th transformation in channel
24442	TRAFO_AXES_IN_6	Axis assignment for the 6th transformation
24444	TRAFO_GEOAX_ASSIGN_TAB_6	Geo-axis assignment for 6th transformation
24450	TRAFO_TYPE_7	Definition of the 7th transformation in channel
24452	TRAFO_AXES_IN_7	Axis assignment for the 7th transformation
24454	TRAFO_GEOAX_ASSIGN_TAB_7	Geo-axis assignment for 7th transformation
24460	TRAFO_TYPE_8	Definition of the 8th transformation in channel
24462	TRAFO_AXES_IN_8	Axis assignment for the 8th transformation
24464	TRAFO_GEOAX_ASSIGN_TAB_8	Geo-axis assignment for 8th transformation
24700	TRAANG_ANGLE_1	Angle of inclined axis in degrees (1st TRAANG)
24710	TRAANG_BASE_TOOL_1	Distance of tool zero point from origin of geometry axes (1st TRAANG)
24720	TRAANG_PARALLEL_VELO_RES_1	Velocity reserve of parallel axis for compensatory motion (1st TRAANG)
24721	TRAANG_PARALLEL_VELO_RES_2	Velocity reserve of parallel axis for compensatory motion (2nd TRAANG)
24750	TRAANG_ANGLE_2	Angle of inclined axis in degrees (2nd TRAANG)

Number	Identifier: $MC_	Description
24760	TRAANG_BASE_TOOL_2	Distance of tool zero point from origin of geometry axes (2nd TRAANG)
24770	TRAANG_PARALLEL_ACCEL_RES_1	Axis acceleration reserve of parallel axis for compensatory motion (1st TRAANG)
24771	TRAANG_PARALLEL_ACCEL_RES_2	Axis acceleration reserve of parallel axis for compensatory motion (2nd TRAANG)

7.11.1.4 Chained transformations

Channel-specific machine data

Number	Identifier: $MC_	Description
24995	TRACON_CHAIN_1	Transformation chain of the first chained transformation
24996	TRACON_CHAIN_2	Transformation chain of the second chained transformation
24997	TRACON_CHAIN_3	Transformation chain of the third chained transformation
24998	TRACON_CHAIN_4	Transformation chain of the fourth chained transformation

7.11.1.5 Non transformation-specific machine data

Channel-specific machine data

Number	Identifier: $MC_	Description
21110	X_AXIS_IN_OLD_X_Z_PLANE	Coordinate system for automatic Frame definition
21090	MAX_LEAD_ANGLE	Maximum permissible lead angle for orientation programming
21092	MAX_TILT_ANGLE	Maximum permissible side angle for orientation programming
21100	ORIENTATION_IS_EULER	Angle definition for orientation programming

7.11.2 Signals

7.11.2.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Transformation active	DB21, ...DBX33.6	DB3300.DBX1.6

M5: Measuring

8

8.1 Brief description

Channel-specific measuring

In the case of channel-specific measuring, a trigger event is programmed for a part program block. This triggers the measuring operation and selects a measuring mode for performing the measurements. The instructions apply to all axes programmed in this particular block.

Preset actual value memory and scratching

The **preset actual value memory** is initiated by means of an HMI operator action.
The calculated frame can be written to system frame $P_SETFRAME. The setpoint position of an axis in the WCS can be altered when the actual value memory is preset.

The calculation is performed in the NC when a PI service is activated via

- HMI operator action or a
- Part program command from the measuring cycles.

The term **scratching** refers to both the workpiece measurement **and** the tool measurement. The measurements can be initiated via the

- HMI operator action or via
- Measuring cycles.

Communication with the NC takes place via predefined system variables.

Workpiece and tool measurement

The position of the workpiece can be measured in relation to an edge, a corner or a hole.

To determine the zero position of the workpiece (workpiece zero W) or a hole, setpoint positions can be added to the measured positions in the WCS. The resulting offsets can be entered in a selected frame.

In the case of tool measurement, the control calculates the distance between the tool tip and the tool carrier reference point T from the tool length specified by the user.

Measuring cycles

A description of how to handle measuring cycles can be found in:
References:
/BNM/ Programming Manual Measuring cycles

8.2 Hardware requirements

8.2.1 Probes that can be used

General information

In order to measure tool and workpiece dimensions, a touch-trigger probe is required that supplies a constant signal (rather than a pulse) when deflected.

The probe must operate virtually bounce-free. Most sensors can be adjusted mechanically to ensure that they operate in this manner.

Different types of probes supplied by a variety of manufacturers are available on the market. Probes are therefore divided into three groups according to the number of directions in which they can be deflected (see figure below).

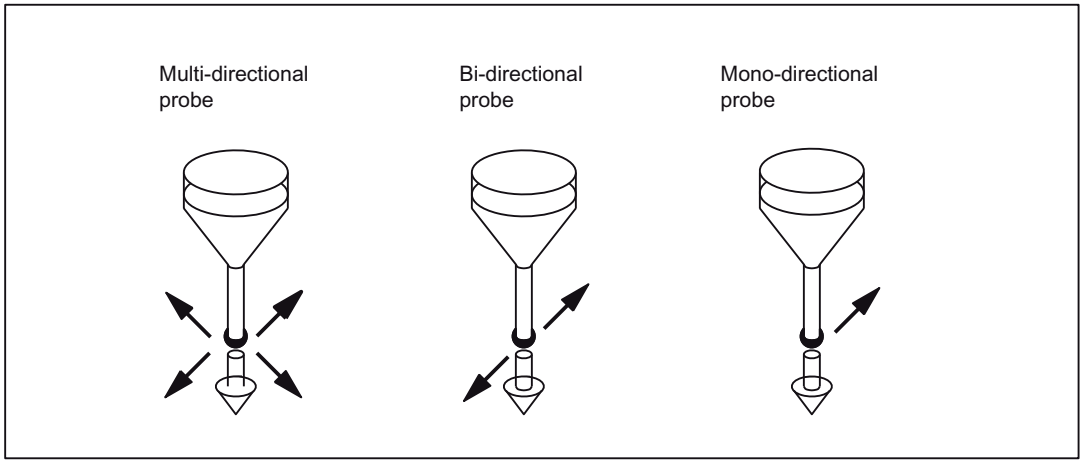


Figure 8-1 Probe types

Table 8- 1 Probe assignment

Probe type	Turning machines		Milling and machining centers
	Tool measurements	Workpiece measurements	Workpiece measurements
Multi-directional	X	X	X
Bi-directional	_	X	X
Mono-directional	_	_	X

While bi-directional probes are used on turning machines for workpiece measurements, a mono-probe can also be used for this purpose on milling and machining centers.

Multidirectional probe (3D)

This probe type can be used unconditionally for measuring tool and workpiece dimensions.

Bidirectional probe

This probe type is handled in the same way as a mono probe in milling and machining centers. Bi-directional probes can be used to take workpiece measurements on turning machines.

Monodirectional probe

This probe type can be used, with only a few restrictions, to take workpiece measurements on milling and machining centers.

Spindle position with mono probe

To be able to use this probe type on milling and machining centers, it must be possible to position the spindle with the NC function SPOS and to transfer the switching signal from the probe over 360° to the receiver station (on the machine stator).

The probe must be mechanically aligned in the spindle to permit measurements in the following directions at the 0 degree position of the spindle.

Table 8- 2 Spindle positions for alignment of probe

	Spindle position measurements at 0 degrees
X-Y plane G17	Positive X direction
Z-X plane G18	Positive Z direction
Y-Z plane G19	Positive Y direction

The measurement takes longer with a mono probe because the spindle needs to be positioned several times with SPOS during the measuring cycle.

8.2.2 Measuring probe connection

Connection to SINUMERIK 840D

The probe is connected to the SINUMERIK 840D system via the I/O device interface X121 located on the front plate of the NCU module.

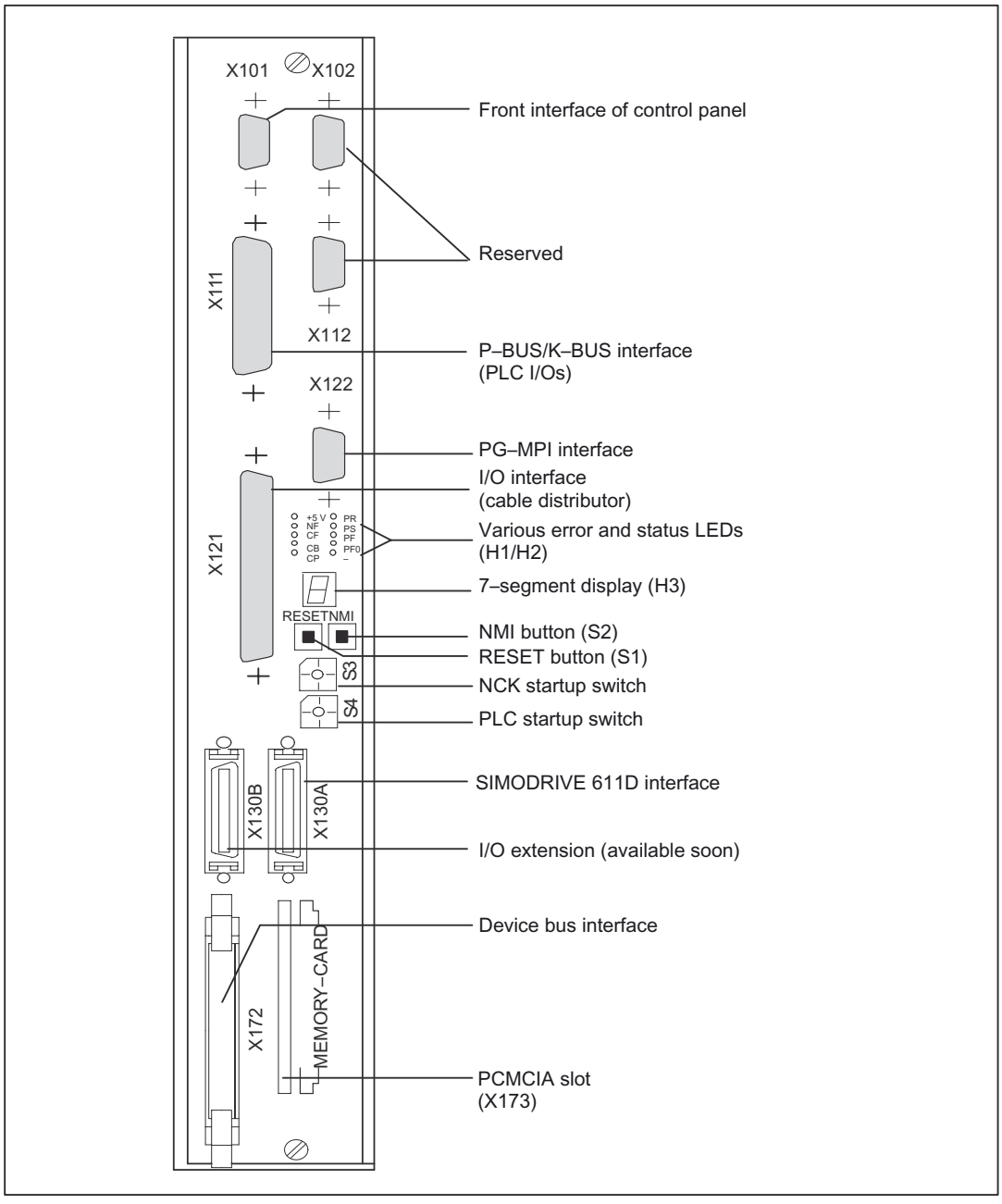


Figure 8-2 Interfaces, control and display elements on the NCU module

Connection to SINUMERIK 840D sl

The probe is connected to the SINUMERIK 840D sl system via the peripheral device interface X121 located on the upper front plate of the NCU module.

Various factory-specific message frame types are programmable for the digital inputs/outputs of this interface.

SINUMERIK 840D sl uses the following preferred message frame types:

- Without probe
- Up to two probes
- Up to six probes

There are two variations of the terminal assignment X122. Each variation is pre-assigned for two probes and both have identical pin assignments. A configuration macro is connected on the line side for both configurable axis probe signals. As a result, the user does not have to carry out any additional probe signal configuration.

References

CNC Commissioning Manual Part 1 (NCK, PLC, drive); Chapter "Basics".
/PHD/ Device Configuration Manual, NCU

Connection to SINUMERIK 840Di

The probe is connected to the SINUMERIK 840Di via the I/O interface X121 on the MCI board extension module (optional).

References:
/HBI/ Configuration Manual, SINUMERIK 840Di

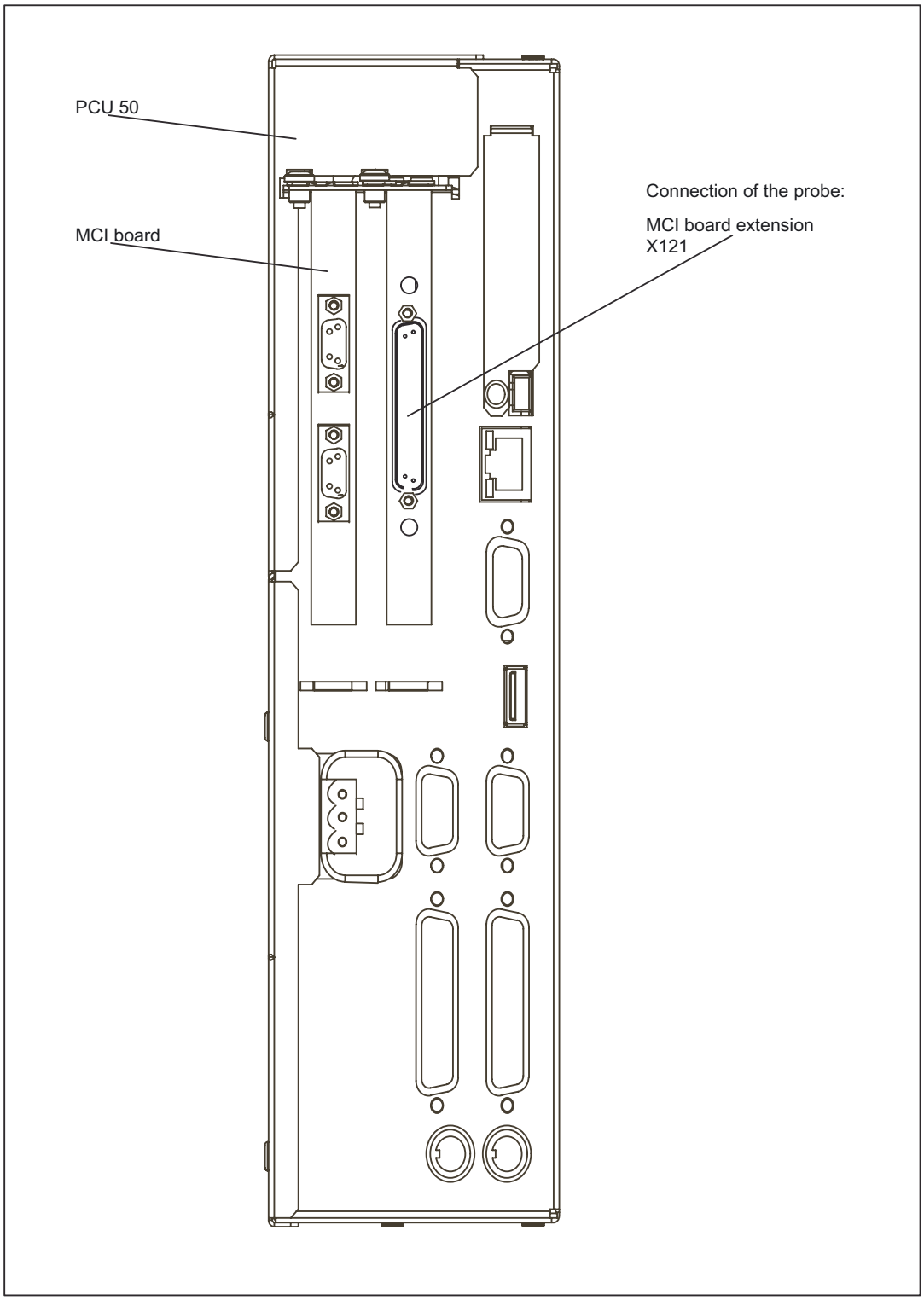


Figure 8-3 SINUMERIK 840Di interfaces (PCU 50, MCI board and MCI board extension)

I/O device interface X121

The interface connection for a probe is made via the

- **I/O device interface**

37-pin D-sub plug connector (X121), a **maximum of 2** probes can be connected;

The 24 V load power supply is also connected by means of this connector.

Table 8- 3 Extract from PIN assignment table for X121 front connectors

	PIN		Name
X121			External power supply
	1	M24EXT	External ground
	2	M24EXT	External ground
	...	...	...
			Probe connection 1
	9	MEPUS 0	Measuring pulse signal input
	10	MEPUC 0	Measuring pulse common input
	...	...	...
			External power supply
	20	P24EXT	P 24 V external
	21	P24EXT	P 24 V external
	...	...	...
			Probe connection 2
	28	MEPUS 1	Measuring pulse signal input
	29	MEPUC 1	Measuring pulse common input
	...	...	...

The interfaces with pin assignment are displayed and described in detail:
References:
/PHC/ CCU Configuration Manual

PROFIBUS-DP drives

It is possible to operate a distributed probe directly on the PROFIBUS-DP drive for the SINUMERK 840D with an NCU 573.2/3/4. This method is more accurate than NC interpolation of cyclic position values from a centralized probe.

The type of measuring function for PROFIBUS-DP drives, e.g., with SIMODRIVE 611 universal, is specified by the following machine data:

MD13210 $MN_MEAS_TYPE

Value = 0: A probe connected centrally to the NC or

Value = 1: A distributed probe wired for **all** drives.

SIMODRIVE 611 universal drives support the measurement functionality of distributed probes by storing the actual encoder value in the hardware concurrent to the measurement signal edge. The more accurate measurement method for a distributed probe is preferred for PROFIBUS-DP drives.

SIMODRIVE 611 digital drives continue to be operated with a centralized probe at connector X121 on the SINUMERIK 840D/840Di.

References:
/BHA/ User Manual, Absolute Value Sensor with PROFIBUS-DP
/FBU/ Function Manual of SIMODRIVE 611 universal

8.3 Channel-specific measuring

8.3.1 Measuring mode

Measuring commands MEAS and MEAW

The measuring operation is activated from the part program. A trigger event and a measuring mode are programmed.

A distinction is made between two measuring modes:

- MEAS: Measurement with deletion of distance-to-go

  Example:

  N10 G01 F300 X300 Z200 MEAS=-2

  Trigger event is the falling edge (-) of the second probe (2).

- MEAW: Measurement without deletion of distance-to-go

  Example:

  N20 G01 F300 X300 Y100 MEAW=1

  Trigger event is the rising edge of the first probe (1).

The measuring job is aborted with `RESET` or when the program advances to a new block.

Note

If a GEO axis is programmed in a measuring block, then the measured values are stored for all current GEO axes.

If an axis participating in a transformation is programmed in a measurement block, the measured values for all axes participating in this transformation are recorded.

Probe status

It is possible to scan the probe status directly in the part program and in synchronized actions.

$A_PROBE[n] where n=probe

$A_PROBE[n]==1: Probe deflected

$A_PROBE[n]==0: Probe not deflected

8.3.2 Measurement results

Read measurement results in PP

The results of the measurement commands are stored in system data of the NCK and can be read via system variables in the part program.

- **System variable $AC_MEA[No]**

  Query measurement job status signal.

  [No] stands for probe (1 or 2)

  The variable is deleted at the beginning of a measurement. The variable is set as soon as the probe fulfills the activation criterion (rising or falling edge). Execution of the measurement job can thus be checked in the part program.

- **System variable $AA_MM[axis]**

  Access to measured value in the machine coordinate system (MCS)

  Read in part program and in synchronized actions.

  [Axis] stands for the name of the measurement axis (X, Y, ...).

- **System variable $AA_MW[axis]**

  Access to measured value in the workpiece coordinate system.

  Read in part program and in synchronized actions.

  [Axis] stands for the name of the measurement axis (X, Y, ...).

References:
/PGZ/ Programming Manual Cycles

PLC service display

The functional test for the probe is performed using an NC program.

The measuring signal can be checked at the end of the program in the diagnostic menu "PLC status".

Table 8- 4 Status display for measurement signal

	Status display
Probe 1 deflected	DB10 DB B107.0
Probe 2 deflected	DB10 DB B107.1

The current measuring status of the axis is displayed by means of the interface signal DB31, ... DBX62.3.

Bit 3=1: Measurement
active Bit 3=0: Measurement not active

This signal can be displayed for all measurement functions and also be read in synchronized actions with

- **system variable $AA_MEAACT[axis]**.

References:

/FBSY/ Function Manual, Synchronized Actions

8.4 Setting zeros, workpiece measuring and tool measuring

8.4.1 Preset actual value memory and scratching

Preset actual value memory

Preset actual value memory is initiated by means of an HMI operator action or via measuring cycles. The calculated frame can be written to system frame $P_SETFRAME. The setpoint position of an axis in the WCS can be altered when the actual value memory is preset.

The calculation is performed in the NC when a PI service is activated via

- HMI operator action or a
- Part program command from the measuring cycles.

A tool and a plane can be selected as a basis for the calculation. The calculated frame is entered in the result frame.

Scratching

The term **scratching** refers to both the workpiece **and** tool measurement. The position of the workpiece can be measured in relation to an edge, a corner or a hole. To determine the zero position of the workpiece or the hole, setpoint positions can be added to the measured positions in the WCS. The resulting offsets can be entered in a selected frame. In the tool measurement, the length or radius of a tool can be measured using a measured reference part.

The measurements can be initiated via

- HMI operator action or via
- Measuring cycles.

Communication with the NC takes place via predefined system variables. The calculation is performed in the NCK when a PI service is activated via:

- the HMI operator action
- or through a part program command from the measuring cycles.

A tool and a plane can be selected as a basis for the calculation. The calculated frame is entered in the result frame.

For more detailed information about channel-specific system frames, please see:

/PGA1/ List of System Variables; chapter "Frames".

/FB1/ Function Manual for Basic Functions, Axes, Coordinate Systems, Frames (K2), Chapter "Frames of the Frame Chain"

Additional references:

/BAD/ BEM/BEMsl Operating Manual, HMI Advanced/Embedded; Chapter "Scratching".

/PGZ/ Programming Manual Cycles; Chapter "Swiveling - CYCLE800"

8.4.2 Workpiece measuring

Workpiece measuring

For workpiece measurement, a probe is moved up to the clamped workpiece in the same way as a tool. Due to the variety of different measuring types available, the most common measurement jobs can be performed quite simply and easily on a turning or milling machine.

The position of the workpiece can be measured in relation to an edge, a corner or a hole.

To determine the zero position of the workpiece (workpiece zero W) or a hole, setpoint positions can be added to the measured positions in the WCS. The resulting offsets can be entered in a selected frame.

Variable interface

The variable interface comprises several system variables,

These are categorized as either:

- Input values
- Output values

References:
/PGA1/ Parameter Manual System Variables

Input values must be written by the HMI or the cycles. The output values result from the calculations.

References:
/BNM/ Measuring Cycles Programming Manual

8.4.2.1 Input values

Validity bits for the measurement types

To define which system variables are valid for the current measurement, each measuring process must first declare all the variables as invalid. This is done with:
$AC_MEAS_VALID = 0.

Each input variable implicitly sets the corresponding bit in $AC_MEAS_VALID when writing to the interface. If the validity bits are not reset, the values remain valid for the next calculation.

Note

The interface is not reset in case of machine control table reset or after M30 (reset at program end).

Table 8- 5 Validity bits for the input values of the variables $AC_MEAS_VALID

Bit	Input value	Description
0	$AA_MEAS_POINT1[axis]	1st measuring point for all channel axes
1	$AA_MEAS_POINT2[axis]	2nd measuring point for all channel axes
2	$AA_MEAS_POINT3[axis]	3rd measuring point for all channel axes
3	$AA_MEAS_POINT4[axis]	4th measuring point for all channel axes
4	$AA_MEAS_SETPOINT[axis]	Setpoint position of the edge, corner, hole
5	$AC_MEAS_WP_SETANGLE	Setpoint workpiece position angle α; -90 < φ < 180
6	$AC_MEAS_CORNER_SETANGLE	Setpoint angle of intersection φ of the corner 0 < φ < 180
7	$AC_MEAS_T_NUMBER	Selected tool
7	$AC_MEAS_D_NUMBER	Selected cutting edge
9	$AC_MEAS_DIR_APPROCH	Approach direction for edge, groove, web and tool measurement only
10	$AC_MEAS_ACT_PLANE	Set working plane and infeed direction
11	$AC_MEAS_FRAME_SELECT	Calculated frame in the specified frame
12	$AC_MEAS_TYPE	Types of workpiece measurement
13	$AC_MEAS_FINE_TRANS	Enter translational offsets
14	$AA_MEAS_SETANGEL[axis]	Setpoint angle of an axis
15	$AA_MEAS_SCALEUNIT	Unit of measurement for input and output values
16	$AA_MEAS_TOOL_MASK	Tool settings
17	$AA_MEAS_P1_COORD	Coordinate system of 1st measuring point
18	$AA_MEAS_P2_COORD	Coordinate system of 2nd measuring point
19	$AA_MEAS_P3_COORD	Coordinate system of 3rd measuring point
20	$AA_MEAS_P4_COORD	Coordinate system of 4th measuring point
21	$AA_MEAS_SET_COORD	Coordinate system of setpoint
22	$AA_MEAS_CHSFR	System frame mask
23	$AA_MEAS_NCBFR	Mask for global basic frame
24	$AA_MEAS_CHBFR	Mask for channel basic frames
25	$AA_MEAS_UIFR	Settable frame from data management
26	$AA_MEAS_PFRAME	Do not calculate programmable frames
27	$AC_MEAS_INPUT[n]	Measuring input parameter with length n

Note

All axis actual values of the appropriate measuring point are invalidated by:

$AC_MEAS_LATCH = 0

Measuring points

A maximum of four measuring points are available for all channel axes for measurement:

Type	Input variable	Description
REAL	$AA_MEAS_POINT1[axis]	1st measuring point for all channel axes
REAL	$AA_MEAS_POINT2[axis]	2nd measuring point for all channel axes
REAL	$AA_MEAS_POINT3[axis]	3rd measuring point for all channel axes
REAL	$AA_MEAS_POINT4[axis]	4th measuring point for all channel axes

The measured points are normally available as actual values (= setpoint values) in WCS. A measuring point is denoted as valid as soon as an axis value is described for it. Each individual measuring point can be written or picked up.

A few measuring types also support measuring points lying in a different coordinates system (BCS, MCS). The entry in which the coordinates system of the corresponding measuring point was measured can be done via the following variables:

Type	Input variable	Description	Values
INT	$AA_MEAS_P1_COORD	Coordinate system of 1st measuring point	0: WCS is the standard setting 1: BCS 2: MCS 3: ENS 4: WCS_REL 5: ENS_REL
INT	$AA_MEAS_P2_COORD	Coordinate system of 2nd measuring point	
INT	$AA_MEAS_P3_COORD	Coordinate system of 3rd measuring point	
INT	$AA_MEAS_P4_COORD	Coordinate system of 4th measuring point	
INT	$AA_MEAS_SET_COORD	Coordinate system of setpoint	

Actual values

The measuring points can be described for all the axes having the current axis actual values. The positions are picked up with reference to the selected coordinates system. The positions are attached in WCS if no coordinates system is specified. The following variable is used for this purpose:
$AC_MEAS_LATCH[0..3]

The index varies from 0 to 3, corresponding to the 1st to 4th measuring point. Assigning the value zero to the variable has the effect that all axis actual values of the corresponding measuring point become invalid. Assigning the value 1 picks up all the axis actual values in the corresponding measuring point. The variable is a write-only variable.

Individual axis actual values of a measuring point can be described with the following variables:

Type	System variable	Description	Values
REAL	$AA_MEAS_P1_VALID[ax]	Pick up the 1st measuring point of an axis	0: The measuring point of the axis is invalid 1: The measuring point of the axis is determined
REAL	$AA_MEAS_P2_VALID[ax]	Pick up the 2nd measuring point of an axis	
REAL	$AA_MEAS_P3_VALID[ax]	Pick up the 3rd measuring point of an axis	
REAL	$AA_MEAS_P4_VALID[ax]	Pick up the 4th measuring point of an axis	

The variables $AC_MEAS_LATCH[0..3] and $AA_MEAS_P[1..4]_VALID can be used interactively. Allowance is made accordingly for the facing axis with diameter programming.

Setpoints

The resultant frame is calculated so that the measurement complies with the setpoints specified by the user.

Table 8- 6 Input values for the user setpoint values

Type	System variable	Description
REAL	$AA_MEAS_SETPOINT[ax]	Setpoint position of an axis
REAL	$AA_MEAS_SETANGLE[ax]	Setpoint angle of an axis
INT	$AA_MEAS_SP_VALID[ax]	1: Setpoint position of axis is valid / 0: Invalid
REAL	$AC_MEAS_WP_SETANGLE	Rated workpiece position angle α: -90 < α < 180
REAL	$AC_MEAS_CORNER_SETANGLE	Setpoint cutting angle φ of corner: 0 < φ < 180
INT	$AC_MEAS_DIR_APPROACH *)	Approach direction: 0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z

*) The approach direction is required only for the edge, groove, web and tool measurement.

The following measuring points are irrelevant and not evaluated:

- On inputting the setpoint workpiece position angle α: of the 2nd measuring point.
- When inputting the setpoint angle of intersection φ: at the 4th measuring point.

Plane separation

Plane separation for defining the tool orientation. The active level is used for all calculations if no level is specified.

Type	System variable	Values
INT	$AC_MEAS_ACT_PLANE	0: G17 working plane x/y infeed direction z 1: G18 working plane z/x infeed direction y 2: G19 working plane y/z infeed direction x

Translational offsets

When measuring workpieces, translational offsets can be entered in the fine offset component of the selected frame. Variable $AC_MEAS_FINE_TRANS is used for this purpose.

Type	System variable	Values
INT	$AC_MEAS_FINE_TRANS	0: Translational compensation is entered in the coarse offset. 1: Translational compensation is entered in the fine offset.

The following is applicable if the variable $AC_MEAS_FINE_TRANS is not described:

- The compensation value is entered in the coarse offset and transformed in the time frame. There can also be a fine portion in the translation by virtue of the transformations.
- If the following machine data is not preset to 1:

  MD18600 $MN_MM_FRAME_FINE_TRANS

  The compensation is always entered in the course offset.

Calculated frame

When a workpiece is measured, the calculated frame is entered in the specified frame.

Type	System variable	Description
INT	$AC_MEAS_FRAME_SELECT	Frame selection during tool measurement

The variable $AC_MEAS_FRAME_SELECT can assume the following values:

Value		Description
0	$P_SETFRAME	Active system frame
1	$P_PARTFRAME	Active system frame
2	$P_EXTFRAME	Active system frame
10..25	$P_CHBFRAME[0..15]	Active channel-specific basic frames
50..65	$P_NCBFRAME[0..15]	active NCU-global basic frames
100..199	$P_IFRAME	The calculation is done using the active settable frame, if the corresponding frame is selected. If the selected frame is not active, the corresponding data management frame is included in the calculation.
500	$P_TOOLFRAME	Active system frame
501	$P_WPFRAME	Active system frame
502	$P_TRAFRAME	Active system frame
503	$P_PFRAME	Active current programmable frame
504	$P_CYCFRAME	Active system frame
505	$P_RELFRAME (PCS)	Active system frame
506	$P_RELFRAME (ACS)	Active system frame
1010..1025	$P_CHBFRAME[0..15]	Active channel specification Basic frames with active G500
1050..1065	$P_NCBFRAME[0..15]	Active NCU-global Basic frames with active G500
2000	$P_SETFR	System frame in data management
2001	$P_PARTFR	System frame in data management
2002	$P_EXTFR	System frame in data management
2010..2025	$P_CHBFR[0..15]	Channel-specific basic frames in data management
2050..2065	$P_NCBFR[0..15]	NCU-global basic frame in data management
2100..2199	$P_UIFR[0..99]	Settable frame in data management
2500	$P_TOOLFR	System frame in data management

Value		Description
2501	$P_WPFR	System frame in data management
2502	$P_TRAFR	System frame in data management
2504	$P_CYCFR	System frame in data management
2505	$P_RELFR (PCS)	System frame in data management
2506	$P_RELFR (ACS)	System frame in data management
3010..3025	$P_CHBFR[0..15]	Channel-spec. Basic frames with active G500 in data management
3050..3065	$P_NCBFR[0..15]	NCU-global basic frames with active G500 in data management

The MEASURE( ) function calculates frame $AC_MEAS_FRAME according to the specified frame.

In the case of values

0 to 1065, the calculation is performed using the active frame.

2000 to 3065, the calculation is performed with reference to the selected frame in data management. The frame selection in data management is not supported for measurement types 14 and 15. A frame does not have to be active in order to select it in data management. In this case, the calculation is performed as if the frame were active in the chain.

The measuring point is transformed in the selected system and the selected frame is determined using the entire frame including the selected frame. Preset actual value memory is active only after compensation and activation of the frame.

In the case of values with active **G500** active (1010..1025, 1050..1065, 3010..3025, 3050..3065), the target frame is calculated so that G500 must be active after the frame is selected so that the setpoint position can be calculated.

Conversion into another coordinate system

If a position is to be converted to a position in another coordinate system, the following variables can be used to specify the composition of the desired frame chain:

Type	System variable	Description	Values
INT	$AC_MEAS_CHSFR	Selection of system frames	Bit mask corresponding to MD28082 $MC_MM_SYSTEM_FRAME_MASK
INT	$AC_MEAS_NCBFR	Selection of global basic frames	Bit mask (0 ... FFFF)
INT	$AC_MEAS_CHBFR	Selection of channel basic frames	Bit mask (0 ... FFFF)
INT	$AC_MEAS_UIFR	Selection of settable frames	0 ... 99
INT	$AC_MEAS_PFRAME	Programmable frame	0: is included 1: is not included

The data management frames are read and a new frame set up for the corresponding values in the variables.

Note

If variables are not set, the active frames are retained.

Values should only be written to those variables whose data management frames are to be included in the new frame chain. In the case of the basic frames, **only all** of the frames can be exchanged, and not just a particular frame. Active changes via $P_NCBFRMASK and $P_CHBFRMASK are ignored.

Array variable for workpiece and tool measurement

The following array variable of length n is used for further input parameters that are used in the various measurement types

Type	System variable	Description	Values
REAL	$AC_MEAS_INPUT[n]	Measurement input parameters	n = 0 ... 9

The control action of the measurement input parameters is described with the measuring methods.

Selection of tool or cutting edge

The tool and edge number of the active tool must correspond to the selected tool. When T0, D0 is selected, the active tool is calculated. If no tool is active, the tool selected by T, D is calculated. No tool other than the selected tool may be active.

Type	System variable	Description
INT	$AC_MEAS_T_NUMBER	Selected tool
INT	$AC_MEAS_D_NUMBER	Selected cutting edge

Measurements with 3D probe

When measuring with the 3D probe, the radius of the tool is already compensated with reference to the measuring point, and therefore the radius does not have to be included when calculating the various measurement operations. This property can be defined by means of the following variable:

Type	System variable	Description
INT	$AC_MEAS_TOOL_MASK	Tool position

The variable $AC_MEAS_TOOL_SCREEN can assume the following values:

Value	Description
0x0	All tool lengths are considered (default setting).
0x1	Tool radius is not included in the calculation
0x2	Tool position in x direction (G19)
0x4	Tool position in y direction (G18)
0x8	Tool position in y direction (G17)
0x10	Tool length is not included in the calculation
0x20	Length of the active tool is included in the coordinate transformation of a position.
0x40	Tool position in x direction.
0x80	Tool position in y direction.
0x100	Tool position in z direction.
0x200	Tool length differential values are included negatively.

Whether or not the radius of a milling tool is included in the calculation can be determined from the tool position and approach direction. If the approach direction is not specified explicitly, it is determined by the selected plane.

Plane	Approach direction
G17	-z direction
G18	- y direction
G19	- x direction

8.4.2.2 Measurement selection

The measurement is selected by means of the following variable:

Type	System variable	Description
INT	$AC_MEAS_TYPE	Measurement type selection

The variable $AC_MEAS_TYPE can assume the following values:

Value		Description
0		Default
1	Edge_x	Measuring the x edge
2	Edge_y	Measuring the y edge
3	Edge_z	Measuring the z edge
4	Corner_1	Measuring Corner 1
5	Corner_2	Measuring Corner 2
6	Corner_3	Measuring Corner 3
7	Corner_4	Measuring Corner 4
8	Hole	Measuring a hole
9	Stud	Measuring a shaft

Value		Description
10 *	Tool length	Measuring the tool length
11 *	ToolDiameter	Measuring the tool diameter
12	Slot	Measuring a groove
13	Plate	Measuring a web
14	Set_Pos	Preset actual value for geometric and special axes
15	Set_AuxPos	Preset actual value memory for special axes only
16	Edge_2P	Measuring an inclined edge
17	Plane_Angles	Angle of a plane
18	Plane_Normal	Angle of a plane with setpoint input
19	Dimension_1	1-dimensional setpoint value
20	Dimension_2	2-dimensional setpoint value
21	Dimension_3	3-dimensional setpoint value
22 *	ToolMagnifier	ShopTurn: Measurement of tool lengths with zoom-in function
23 *	ToolMarkedPos	ShopTurn: Measuring a tool length with marked position
24	Coordinate transformation	Coordinate transformation of a position
25	Rectangle	Measurement of a rectangle
26	Save	Saving data management frames
27	Restore	Restoring data management frames
28	Taper turning	Additive rotation of the plane

* Types of workpiece measurement

The individual methods are listed under "Types of workpiece measurement" or "Types of tool measurement"and explained in more detail using an appropriate programming example.

8.4.2.3 Output values

Calculation results

If a setpoint position has been specified, the resulting frame is entered in result frame $AC_MEAS_FRAME. This frame can be read and written in the part program. The result frame is calculated according to the selected frame.

If no frame has been selected, the result frame determines the final translation and rotation in the WCS. This frame can be entered in the selected frame using the PI service _N_SETUDT and parameter type no. 7. Once it has been entered, the result frame is deleted.

Table 8- 7 Output values of calculation results

Type	System variable	Description
FRAME	$AC_MEAS_FRAME	Result frame
REAL	$AC_MEAS_WP_ANGLE	Calculated workpiece position angle α
REAL	$AC_MEAS_CORNER_ANGLE	Calculated angle of intersection ϕ

Type	System variable	Description
REAL	$AC_MEAS_DIAMETER	Calculated diameter
REAL	$AC_MEAS_TOOL_LENGTH	Calculated tool length
REAL	$AC_MEAS_RESULTS[10]	Calculation results (depending on $AC_MEAS_TYPE)

8.4.2.4 Calculation method

Activating the calculation

The calculation is activated by an HMI operator action with PI service _N_SETUDT. This PI service can accept one of the following parameter types:

Type	Description
1	Active tool offset
2	Active basic frames
3	Active settable frame
4	Global basic frames
5	Globally settable frames
6	Calculating workpiece zero point or tool lengths
7	Activate workpiece zero point (write scratching)
8	Activate external work offset
9	Activate active tool carrier, TCOABS and PAROT

The modification becomes apparent immediately in the reset state; in the stop state, the frame is not applied until the next start.

Note

The PI service can be executed only in the reset and stop states. In the case of workpiece measurement, the calculated frame is activated immediately with type no. 7. In the case of tool measurement, the PI must not be dispatched with type no. 7, since a zero point does not have to be activated.

Activation in the Stop state

The new WCS positions are refreshed in the Stop state. When execution of the part program is resumed, the distance to go in the interrupted block is deleted and the axis approaches the end point of the next block from its current position.

Therefore, it is possible to start a spindle in MDA mode or in part program and execute preset actual values and scratching or execute another measurement, e.g. with M0, in the part program in the Stop state.

Measuring cycles

The calculation in the measuring cycles is performed according to the predefined function:

INT MEASURE( )

MEASURE() delivers a result frame that can be read via $AC_MEAS_FRAME:

- The result is the translation and rotation from the setpoint values recalculated on the selected frame.
- The result frame is calculated as follows:

  The concatenated total frame equals the concatenation of the total frame (prior to measurement) with the calculated translation and rotation.

Note

If no frame is selected, the calculated frame is not transformed, i.e. the translation and rotation is determined on the basis of the specified setpoints and the calculated position of the edge, corner, groove, etc. Where the function is used more than once, it is always added to the result frame.

It must be noted that the result frame may need to be deleted beforehand.

NOTICE
MEASURE() does not trigger any implicit block search stop. As MEASURE() works with the frames of the block search set, it must itself decide whether a block search stop is necessary before the calculation.

Semaphore variable

The measurement variable occurs only once per channel. The measuring operation can be initiated via an operator input in the stop and reset states. The operation can overlap with the measuring cycles in the stop state. The following variable serves the purpose of protection of mutual overwriting:

$AC_MEAS_SEMA (Semaphore of measurement interface)

The semaphore variable $AC_MEAS_SEMA is

- set to 1 at the beginning of the cycle and
- reset to 0 again at the end of the cycle.

HMI does not use the measurement interface if the variable has the value 1.

Error messages

If the client does not log on, group error number 0xD003 is always generated. If a logon takes place through DIAGN:errCodeSetNrGent or DIAGN:errCodeSetNrPi, then PI_SETUDT provides the error code corresponding to the following syntax:

EX_ERR_PI_REJ_<Return value>, e.g.: EX_ERR_PI_REJ_ MEASNOTYPE

The following return values are output via the pre-defined MEASURE() function:

Table 8- 8 Predefined error messages

No.	Return values	Description
0	MEAS_OK	Correct calculation
1	MEAS_NO_TYPE	Type not specified
2	MEAS_TOOL_ERROR	Error determining the tool
3	MEAS_NO_POINT1	Measuring point 1 does not exist
4	MEAS_NO_POINT2	Measuring point 2 does not exist
5	MEAS_NO_POINT3	Measuring point 3 does not exist
6	MEAS_NO_POINT4	Measuring point 4 does not exist
7	MEAS_NO_SPECPOINT	No reference point available
8	MEAS_NO_DIR	No approach direction
9	MEAS_EQUAL_POINTS	Measuring points are identical
10	MEAS_WRONG_ALPHA	Alpha α is wrong
11	MEAS_WRONG_PHI	Phi ϕ is wrong
12	MEAS_WRONG_DIR	Wrong approach direction
13	MEAS_NO_CROSSING	Lines do not intersect
14	MEAS_NO_PLANE	Planes do not exist
15	MEAS_WRONG_FRAME	No frame or incorrect frame selected
16	MEAS_NO_MEMORY	Insufficient memory available
17	MEAS_INTERNAL_ERROR	Internal error

Tool measurement error

In the case of error code MEAS_TOOL_ERROR or EX_ERR_PI_REJ_MEASTOOLERROR, the system stores a more detailed specification of the error with the following values in output variable $AC_MEAS_TOOL_LENGTH:

Table 8- 9 Predefined error messages for MEAS_TOOL_ERROR

No.	Return values	Description
1	TOOL_NO_BLOCK	No block available for the tool calculation
2	TOOL_WRONG_T_NUMBER	Wrong T number
3	TOOL_WRONG_D_NUMBER	Wrong D number
4	TOOL_EVAL_WRONG_TYPE	The tool does not exist
5	TOOL_NO_TOOLCORR_BODY	Memory problem
6	TOOL_DATA_READ_ERROR	Error reading the tool data
7	TOOL_NO_TOOL_WITH_TRAFO	No tool is selected with an active transformation

8.4.2.5 Units of measurement and measurement variables for the calculation

INCH or METRIC unit of measurement

The following input and output variables are evaluated with inch or metric units of measurement:

$AA_MEAS_POINT1[axis]	Input variable for 1st measuring point
$AA_MEAS_POINT2[axis]	Input variable for 2nd measuring point
$AA_MEAS_POINT3[axis]	Input variable for 3rd measuring point
$AA_MEAS_POINT4[axis]	Input variable for 4th measuring point
$AA_MEAS_SETPOINT[axis]	Input variable for setpoint position
$AC_MEAS_DIAMETER	Output variable for calculated diameter
$AC_MEAS_TOOL_LENGTH	Output variable for calculated tool length
$AC_MEAS_RESULTS[n]	Output variable for calculation results

The system of units in which the input and output values can be read or written can be set via the input variable.

INT $AC_MEAS_SCALEUNIT	Unit of measurement for input and output variable
0: Unit of measurement with reference to	active G codes G70/G700 in INCH active G codes G71/G701 in METRIC
1: Unit of measurement corresponds to	the configuration; the measurement system can be set via OPI (standard setting)

The value 1 is always treated as the standard setting if the variable is not written.

Example:

The basic system is metric:

```
G70
$AC_MEAS_POINT1[x] = $AA_IW[x]            ; $AA_IW[x] supplies the basic system
$AC_MEAS_POINT1[x] = 10                   ; 10 mm
G71
$AC_MEAS_POINT1[x] = $AA_IW[x]            ; $AA_IW[x] supplies the basic system
$AC_MEAS_POINT1[x] = 10                   ; 10 mm
G700
$AC_MEAS_POINT1[x] = $AA_IW[x]            ; $AA_IW[x] supplies inch value
$AC_MEAS_POINT1[x] = 10                   ; 10 inch
G710
$AC_MEAS_POINT1[x] = $AA_IW[x]            ; $AA_IW[x] supplies metric value
$AC_MEAS_POINT1[x] = 10                   ; 10 mm
```

Diameter programming

Diameter programming is set via machine data:

```
MD20100 $MC_DIAMETER_AX_DEF = "X"              ; Transverse axis is x
MD20150 $MC_GCODE_RESET_VALUES[28] = 2         ; DIAMON
MD20360 $MC_TOOL_PARAMETER_DEF_MASK            ; Tool length, frames and
= 'B1001010'                                   ; Actual value in the diameter
```

The following is to be taken into account:

- Axis positions in the MCS are not included as diameter value.
- The calculated tool lengths and frame components do not depend on the active G code DIAMON or DIAMOF.
- The measured positions and setpoint positions are read and written depending on DIAMON.
- The translations in the frames are calculated as a diameter in the transverse axis.

Arithmetic and display precision

Position values in mm, inches or degrees are accurately calculated and displayed to six decimal places.

8.4.2.6 Diagnostics

The following diagnostic options exists for the measurement interface:

- If the file /_N_MPF_DIR/_N_MEAS_DUMP_MPF is available, a log is written in the file that should enable the reproduction of the problem.
- The logging is started by creating a blank file having the filename _N_MEAS_DUMP_MPF in the /_N_MPF_DIR directory.
- The content of the file is preserved till it is deleted with $AC_MEAS_VALID = 0.

For runtime reasons, the trace should be activated only if a problem is detected.

8.4.3 Types of workpiece measurement

8.4.3.1 Measurement of an edge (measurement type 1, 2, 3)

Measurement of an x edge ($AC_MEAS_TYPE = 1)

The edge of a clamped workpiece is measured by approaching this edge with a known tool.

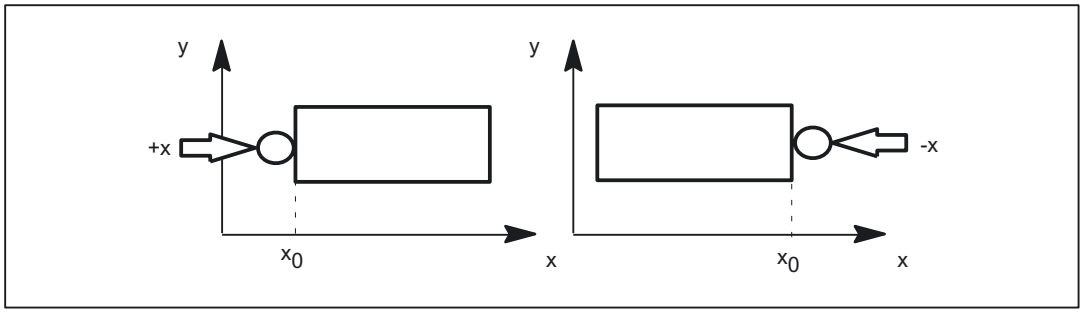


Figure 8-4 x edge

The values of the following variables are evaluated for measurement type 1:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of x edge *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x
$AC_MEAS_ACT_PLANE	Without specification, calculation is undertaken with the active plane, the radius of the tool is used only in G17 and G18 *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	1

* optional

The following output variables are written for measurement type 1:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translations
$AC_MEAS_RESULTS[0]	Position of the measured edge

Example

x edge measurement

```
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                        ; Type
$TC_DP2[1,1]=20                         ; 0
$TC_DP3[1,1]= 10                        ; (z) length compensation vector
$TC_DP4[1,1]= 0                         ; (y)
$TC_DP5[1,1]= 0                         ; (x)
$TC_DP6[1,1]= 2                         ; Radius

T1 D1
g0 x0 y0 z0 f10000
G54

                                        ; Measure x edge
$AC_MEAS_VALID = 0                      ; Set all input values to invalid

g1 x-1 y-3                              ; Approach 2st measuring point

$AA_MEAS_POINT1[x] = $AA_IW[x]
$AA_MEAS_POINT1[y] = $AA_IW[y]
$AA_MEAS_POINT1[z] = $AA_IW[z]

$AC_MEAS_DIR_APPROACH = 0               ; Set approach direction +x

$AA_MEAS_SETPOINT[x] = 0                ; Set setpoint position of the edge
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0

$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17

$AC_MEAS_FRAME_SELECT = 101             ; Select frame - IFRAME

$AC_MEAS_T_NUMBER = 1                   ; Select tool
$AC_MEAS_D_NUMBER = 1

$AC_MEAS_TYPE = 1                       ; Set measurement type for x edge

RETVAL = MEASURE()                      ; Start measuring process

if RETVAL <> 0
```

```
setal(61000 + RETVAL)
endif

$P_IFRAME = $AC_MEAS_FRAME

$P_UIFR[1] = $P_IFRAME                ; Describe system frame in data management

g1 x0 y0                              ; Approach the edge

m30
```

Measurement of a y edge ($AC_MEAS_TYPE = 2)

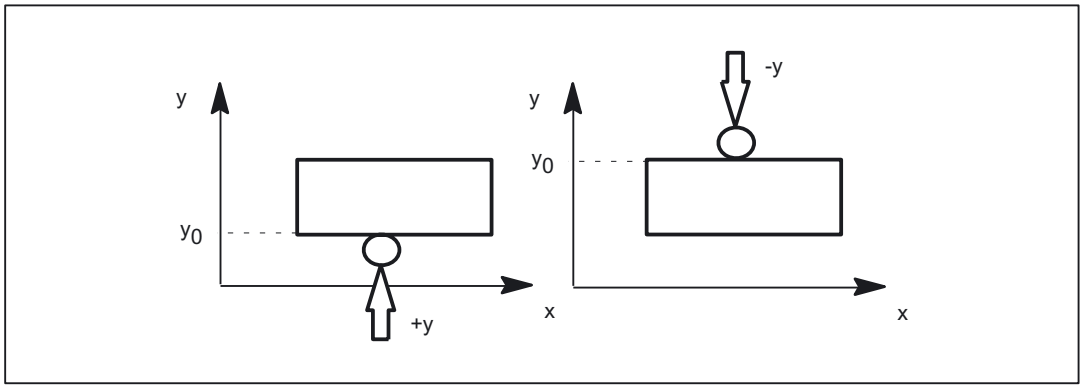


Figure 8-5 y edge

The values of the following variables are evaluated for measurement type 2:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of y edge *
$AC_MEAS_DIR_APPROACH	2: +y, 3: -y
$AC_MEAS_ACT_PLANE	Without specification, calculation is undertaken with the active plane, the radius of the tool is used only in G17 and G19 *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	2

* optional

The following output variables are written for measurement type 2:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of the measured edge

Measurement of a z edge ($AC_MEAS_TYPE = 3)

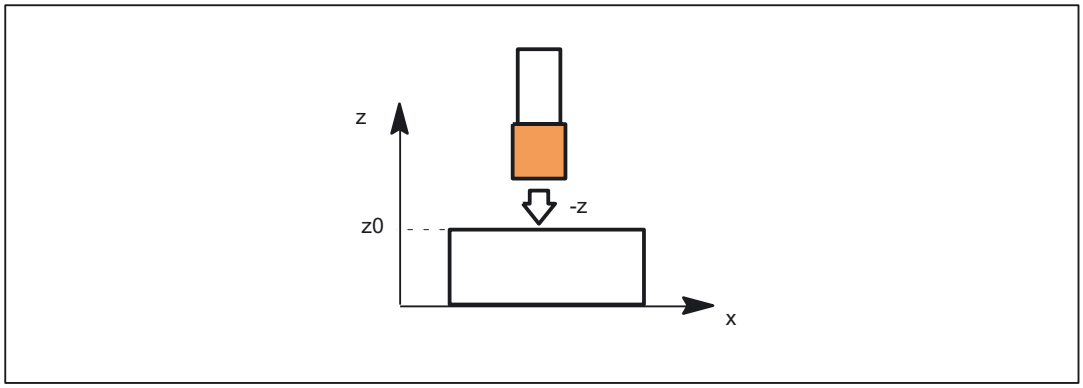


Figure 8-6 z edge

The values of the following variables are evaluated for measurement type 3:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of z edge *
$AC_MEAS_DIR_APPROACH	4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Without specification, calculation is undertaken with the active plane, the radius of the tool is used only in G18 and G19 *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	3

* optional

The following output variables are written for measurement type 3:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of the measured edge

8.4.3.2 Measurement of an angle (measurement type 4, 5, 6, 7)

Measurement of a corner C1 - C4 ($AC_MEAS_TYPE = 4, 5, 6, 7)

A corner is uniquely defined by approaching four measuring points P1 to P4.
Three measurement points suffice in the case of known angles of intersection ϕ.

If the angle of intersection ϕ and the workpiece position angle α are known, two measurement points P1 and P3 suffice.

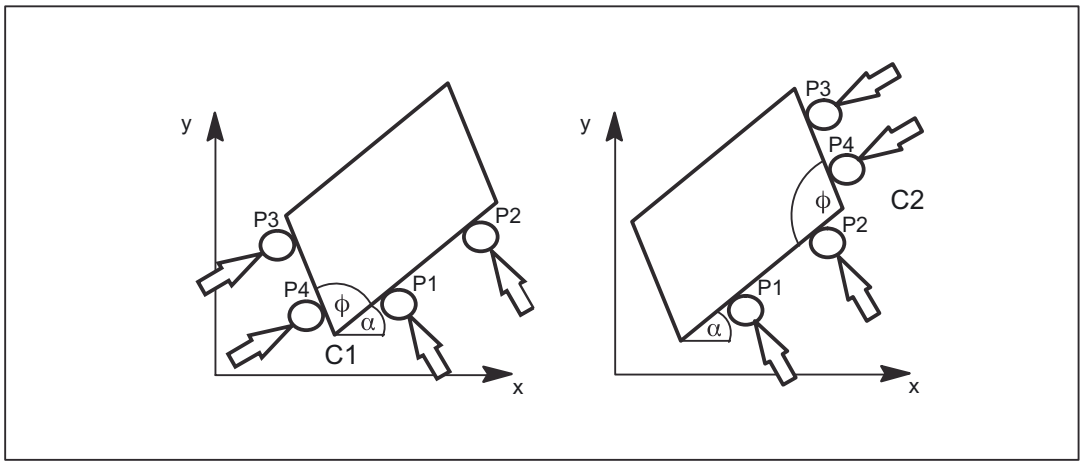


Figure 8-7 Corner C1, corner C2

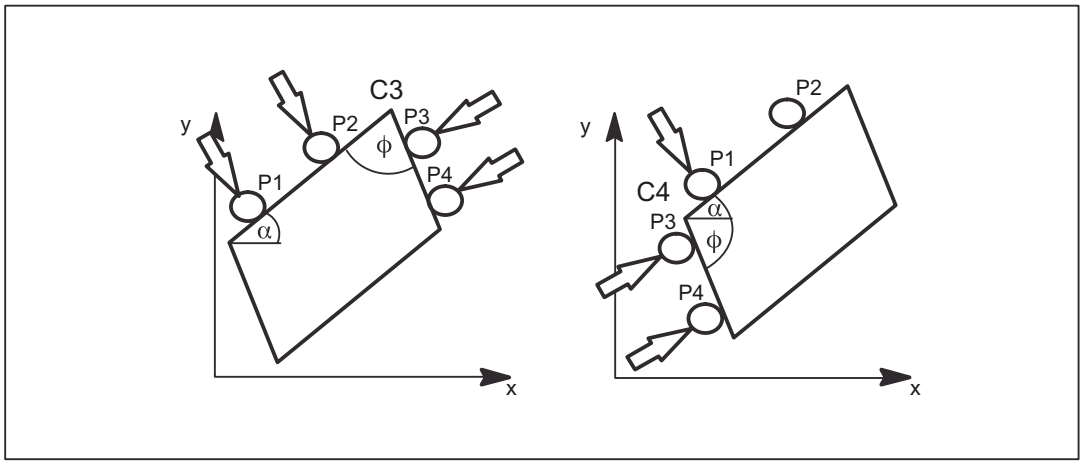


Figure 8-8 Corner C3, corner C4

The values of the following variables are evaluated for measurement types 4 to 7:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2 irrelevant for $AC_MEAS_WP_SETANGLE
$AA_MEAS_POINT3[axis]	Measuring point 3

Input variable	Description
$AA_MEAS_POINT4[axis]	Measuring point 4 irrelevant for $AC_MEAS_CORNER_SETANGLE
$AA_MEAS_WP_SETANGLE	Setpoint workpiece position angle *
$AA_MEAS_CORNER_SETANGLE	Setpoint angle of intersection *
$AA_MEAS_SETPOINT[axis]	Setpoint position of corner *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Without specification of outer corner * =0: Measurement for outer corner =1: Measurement for inner corner
$AC_MEAS_TYPE	4, 5, 6, 7

* optional

The following variables are written for measurement types 4 to 7:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translation and rotation
$AC_MEAS_WP_ANGLE	Calculated workpiece position angle
$AC_MEAS_CORNER_ANGLE	Calculated angle of intersection
$AC_MEAS_RESULTS[0]	Abscissa of calculated vertex
$AC_MEAS_RESULTS[1]	Ordinate of calculated vertex
$AC_MEAS_RESULTS[2]	Applicate of calculated vertex

Example

Corner measurement C1: Corner with three measuring points (P1, P3 and P4) and known angle of intersection ϕ (90°) and unknown workpiece position angle α.

```
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                        ; Type
$TC_DP2[1,1]=20                         ; 0
$TC_DP3[1,1]= 10                        ; (z) length compensation vector
$TC_DP4[1,1]= 0                         ; (y)
$TC_DP5[1,1]= 0                         ; (x)
$TC_DP6[1,1]= 2                         ; Radius

T1 D1
g0 x0 y0 z0 f10000
G54
```

```
$P_CHBFRAME[0] = crot(z,45)
$P_IFRAME[x,tr] = -sin(45)
$P_IFRAME[y,tr] = -sin(45)
$P_PFRAME[z,tr] = -45

                                        ; Measure corner with 3 measuring points
$AC_MEAS_VALID = 0                      ; Set all input values to invalid

g1 x-1 y-3                              ; Approach 1st measuring point
$AC_MEAS_LATCH[0] = 1                   ; Pick up measuring point P1

g1 x-4 y4                               ; Approach 3rd measuring point
$AC_MEAS_LATCH[2] = 1                   ; Pick up measuring point P3

g1 x-4 y1                               ; Approach 4th measuring point
$AC_MEAS_LATCH[3] = 1                   ; Pick up measuring point P4

$AA_MEAS_SETPOINT[x] = 0                ; Set position setpoint of the corner to
                                          (0, 0, 0)
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0

$AC_MEAS_CORNER_SETANGLE = 90           ; Define setpoint angle of intersection φ
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 0               ; Select frame - SETFRAME

$AC_MEAS_T_NUMBER = 1                   ; Select tool
$AC_MEAS_D_NUMBER = 1

$AC_MEAS_TYPE = 4                       ; Set measuring type on corner 1

RETVAL = MEASURE()                      ; Start measuring process

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_CORNER_ANGLE <> 90          ; Query known setpoint angle of
                                          intersection φ
setal(61000 + $AC_MEAS_CORNER_ANGLE)
endif

$P_SETFRAME = $AC_MEAS_FRAME
```

```
$P_SETFR = $P_SETFRAME                    ; Describe system frame in data management

g1 x0 y0                                  ; Approach the corner

g1 x10                                    ; Approach the rectangle
y10
x0
y0

m30
```

8.4.3.3 Measurement of a hole (measurement type 8)

Measuring points for determining a hole ($AC_MEAS_TYPE = 8)

Three measuring points are needed to determine the center point and diameter. The three points must all be different. With specification of four points, the circle is adjusted in accordance with the least square method. The circle is determined so that the sum of the distance squares of the points to the circle is minimal. The quality of the adjustment can be read.

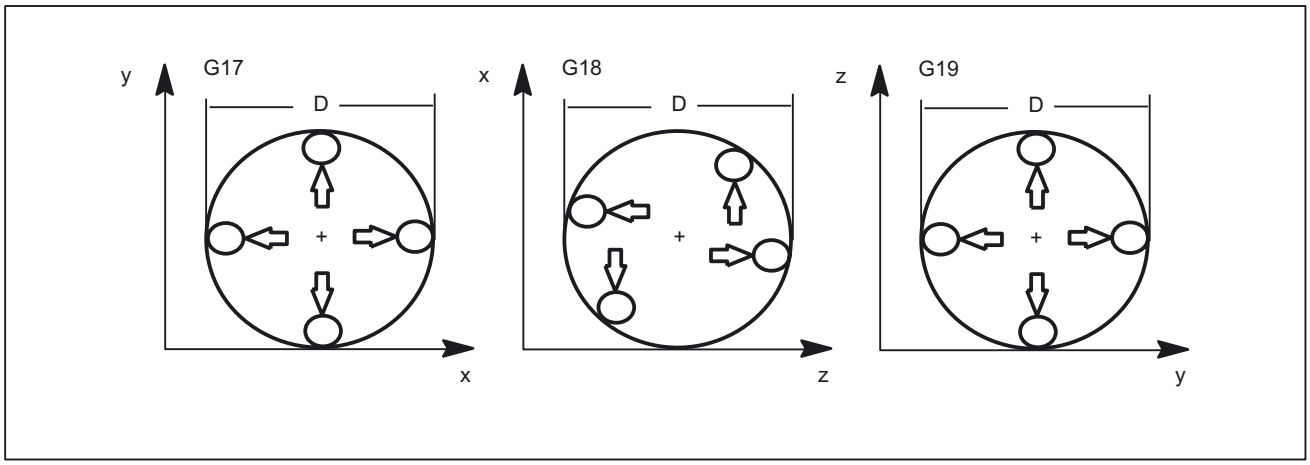


Figure 8-9 Hole

The values of the following variables are evaluated for measurement type 8:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_POINT4[axis]	When specified, the center is determined from four points *
$AA_MEAS_SETPOINT[axis]	Setpoint position of hole center *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *

Input variable	Description
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	8

* optional

The following output variables are written for measurement type 8:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_DIAMETER	Diameter of hole
$AC_MEAS_RESULTS[0]	Abscissa of the calculated center point
$AC_MEAS_RESULTS[1]	Ordinate of the calculated center point
$AC_MEAS_RESULTS[2]	Applicate of the calculated center point
$AC_MEAS_RESULTS[3]	Quality of the circle adjustment: Sum of the distance squares

Example

Measuring a hole

```
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                    ;  Type
$TC_DP2[1,1]=20                     ;  0
$TC_DP3[1,1]= 10                    ;  (z) length compensation vector
$TC_DP4[1,1]= 0                     ;  (y)
$TC_DP5[1,1]= 0                     ;  (x)
$TC_DP6[1,1]= 2                     ;  Radius

T1 D1
g0 x0 y0 z0 f10000
G54

                                    :  Measure hole
$AC_MEAS_VALID = 0                  ;  Set all input values to invalid

g1 x-3 y0                           ;  Approach 1st measuring point
$AA_MEAS_POINT1[x] = $AA_IW[x]
$AA_MEAS_POINT1[y] = $AA_IW[y]
$AA_MEAS_POINT1[z] = $AA_IW[z]
```

```
g1 x0 y3                          ;  Approach 2nd measuring point
$AA_MEAS_POINT2[x] = $AA_IW[x]
$AA_MEAS_POINT2[y] = $AA_IW[y]
$AA_MEAS_POINT2[z] = $AA_IW[z]

g1 x3 y0                          ;  Approach 3rd measuring point
$AA_MEAS_POINT3[x] = $AA_IW[x]
$AA_MEAS_POINT3[y] = $AA_IW[y]
$AA_MEAS_POINT3[z] = $AA_IW[z]

$AA_MEAS_SETPOINT[x] = 0          ;  Set setpoint position of the center
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0

$AC_MEAS_ACT_PLANE = 0            ;  Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 0         ;  Select frame - SETFRAME

$AC_MEAS_T_NUMBER = 1             ;  Select tool
$AC_MEAS_D_NUMBER = 1

$AC_MEAS_TYPE = 8                 ;  Set measuring type on hole

RETVAL = MEASURE()                ;  Start measuring process
if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_DIAMETER <> 10        ;  Query known diameter
setal(61000 + $AC_MEAS_WP_ANGLE)
endif

$P_SETFRAME = $AC_MEAS_FRAME

$P_SETFR = $P_SETFRAME            ;  Describe system frame in data management

g1 x-3 y0                         ;  Approach P1

g2 I = $AC_MEAS_DIAMETER / 2      ;  Approach hole in reference to the center of
                                     the circle


m30
```

8.4.3.4 Measurement of a shaft (measurement type 9)

Measuring points for determining a shaft ($AC_MEAS_TYPE = 9)

Three measuring points are needed to determine the center point and diameter. The three points must all be different. When four points are specified, the circle is adjusted in accordance with the least square method. The circle is determined so that the sum of the distance squares of the points to the circle is minimal. The quality of the adjustment can be read.

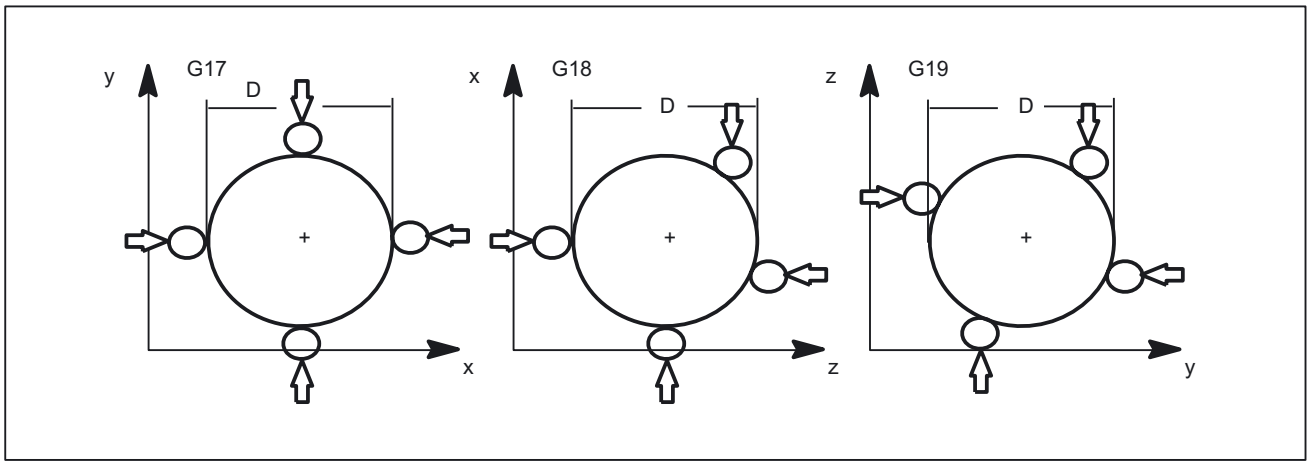


Figure 8-10 Shaft

The values of the following variables are evaluated for measurement type 9:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_POINT4[axis]	When specified, the center is determined from four points *
$AA_MEAS_SETPOINT[axis]	Setpoint position of shaft center point *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	9

* optional

The following output variables are written for measurement type 9:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_DIAMETER	Diameter of hole
$AC_MEAS_RESULTS[0]	Abscissa of the calculated center point
$AC_MEAS_RESULTS[1]	Ordinate of the calculated center point
$AC_MEAS_RESULTS[2]	Applicate of the calculated center point
$AC_MEAS_RESULTS[3]	Quality of the circle adjustment: Sum of the distance squares

8.4.3.5 Measurement of a groove (measurement type 12)

Measuring points for determining the position of a groove ($AC_MEAS_TYPE = 12)

A groove is measured by approaching the two outside corners or inner edges. The groove center can be set to a setpoint position. The component of the approach direction determines the groove position.

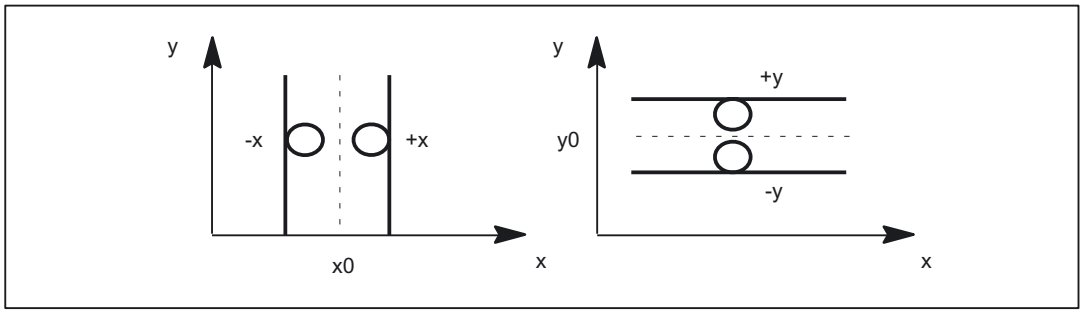


Figure 8-11 Groove

The values of the following variables are evaluated for measurement type 12:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_SETPOINT[axis]	Setpoint position of groove center *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *

Input variable	Description
$AC_MEAS_INPUT[0]	Approach direction for 2nd measuring point for a recess measurement. Must have the same coordinate as the approach direction of the 1st point. * 0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_TYPE	12

* optional

The following output variables are written for measurement type 12:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of calculated groove center (x0, y0 or z0)
$AC_MEAS_RESULTS[1]	Groove width in approach direction

Example

Groove measurement with approach direction in x

```
DEF INT RETVAL
DEF FRAME TMP

$TC_DP1[1,1]=120                    ; Type
$TC_DP2[1,1]=20                     ; 0
$TC_DP3[1,1]= 10                    ; (z) length compensation vector
$TC_DP4[1,1]= 0                     ; (y)
$TC_DP5[1,1]= 0                     ; (x)
$TC_DP6[1,1]= 2                     ; Radius

T1 D1
g0 x0 y0 z0 f10000
G54

$P_CHBFRAME[0] = crot(z,45)
$P_IFRAME[x,tr] = -sin(45)
$P_IFRAME[y,tr] = -sin(45)
$P_PFRAME[z,rt] = -45

                                    ; Measure groove
$AC_MEAS_VALID = 0                  ; Set all input values to invalid

g1 x-2                              ; Approach 1st measuring point
$AA_MEAS_POINT1[x] = $AA_IW[x]
$AA_MEAS_POINT1[y] = $AA_IW[y]
$AA_MEAS_POINT1[z] = $AA_IW[z]
```

```
g1 x4                                ; Approach 2nd measuring point
$AA_MEAS_POINT2[x] = $AA_IW[x]
$AA_MEAS_POINT2[y] = $AA_IW[y]
$AA_MEAS_POINT2[z] = $AA_IW[z]

$AA_MEAS_SETPOINT[x] = 0             ; Set setpoint position of the groove center
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0

$AC_MEAS_DIR_APPROACH = 0            ; Set approach direction +x
$AC_MEAS_ACT_PLANE = 0               ; Measuring plane is G17
$AC_MEAS_FRAME_SELECT = 0            ; Select frame - SETFRAME

$AC_MEAS_T_NUMBER = 1                ; Select tool
$AC_MEAS_D_NUMBER = 1

$AC_MEAS_TYPE = 12                   ; Set measuring type on groove

RETVAL = MEASURE()                   ; Start measuring process

if RETVAL <> 0 setal(61000 + RETVAL)
endif

$P_SETFRAME = $AC_MEAS_FRAME
$P_SETFR = $P_SETFRAME               ; Describe system frame in data management

g1 x0 y0                             ; Approach the groove center

m30
```

8.4.3.6 Measurement of a web (measurement type 13)

Measuring points for determining the position of a web ($AC_MEAS_TYPE = 13)

A web is measured by approaching the two outside corners or inner edges. The web center can be set to a setpoint position. The component of the approach direction determines the web position.

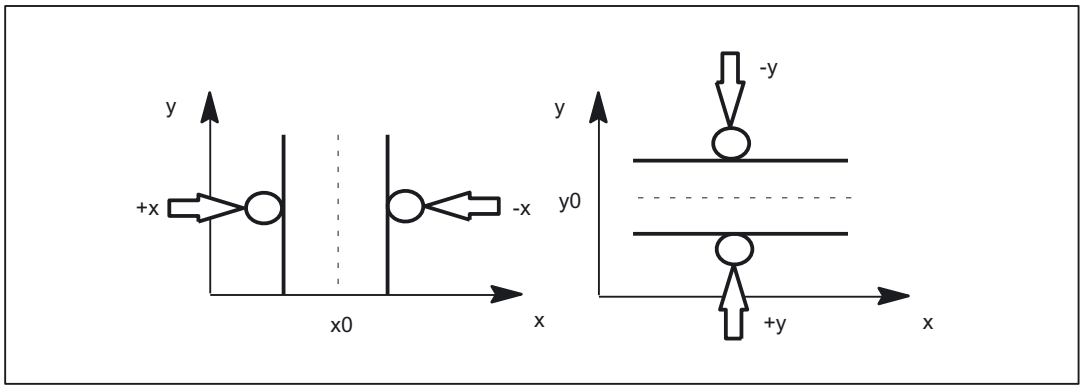


Figure 8-12 Web

The values of the following variables are evaluated for measurement type 13:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_SETPOINT[axis]	Setpoint position of web center *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Approach direction for 2nd measuring point for a recess measurement. Must have the same coordinate as the approach direction of the 1st point. * 0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_TYPE	13

* optional

The following output variables are written for measurement type 13:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Position of calculated web center (x0, y0 or z0)
$AC_MEAS_RESULTS[1]	Web width in approach direction

8.4.3.7 Measurement of geo axes and special axes (measurement type 14, 15)

Preset actual value memory for geo axes and special axes ($AC MEAS TYPE = 14)

This measurement type is used on the HMI operator interface.

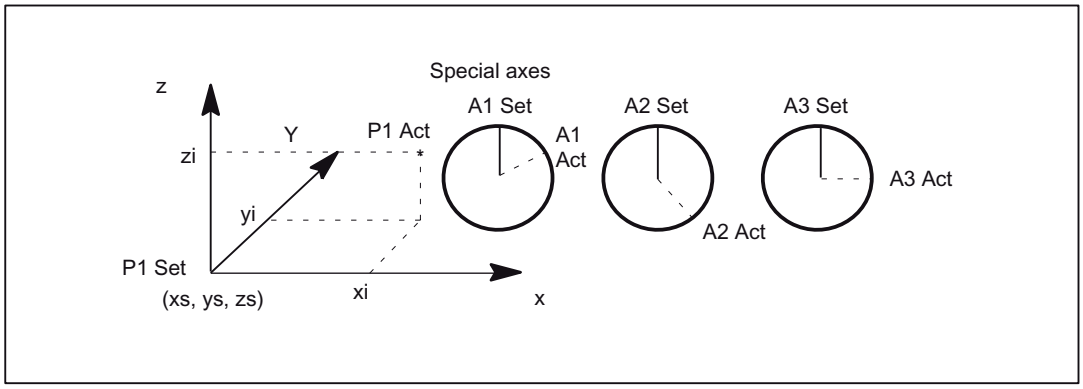


Figure 8-13 Preset actual value memory

The values of the following variables are evaluated for measurement type 14:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Actual values of the axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of the individual axes *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	14

* optional

The following output variables are written for measurement type 14:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translation

Example:

Reference point setting in relative coordinate systems.

```
DEF INT RETVAL

T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54

TRANS x=10                              ; Offset between WCS and ENS
G0 x0 f10000                            ; WCS(x) = 0; ENS(x) = 10

$AC_MEAS_VALID = 0                      ; Set all input variables to invalid
$AC_MEAS_TYPE = 14                      ; Measuring type for preset actual value
                                          memory
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17

$AC_MEAS_P1_COORD = 5                   ; ENS_REL for 1st measuring point
$AC_MEAS_LATCH[0] = 1                   ; Pick up all axis positions

$AC_MEAS_SET_COORD = 5                  ; Setpoint position is relative to ENS
$AA_MEAS_SETPOINT[x] = 0                ; Setpoint position in the relative ENS
                                          coordinate system
$AC_MEAS_FRAME_SELECT = 2505            ; $P_RELFR

RETVAL = MEASURE()                      ; Calculation of $P_RELFR; PI SETUDT(6)
IF RETVAL <> 0 GOTOF ERROR
ENDIF $ P_RELFR = $AC_MEAS_FRAME        ; Activation; PI SETUDT(7)
```

Preset actual value memory for special axes only ($AC MEAS TYPE = 15)

This measurement type is used on the HMI operator interface.

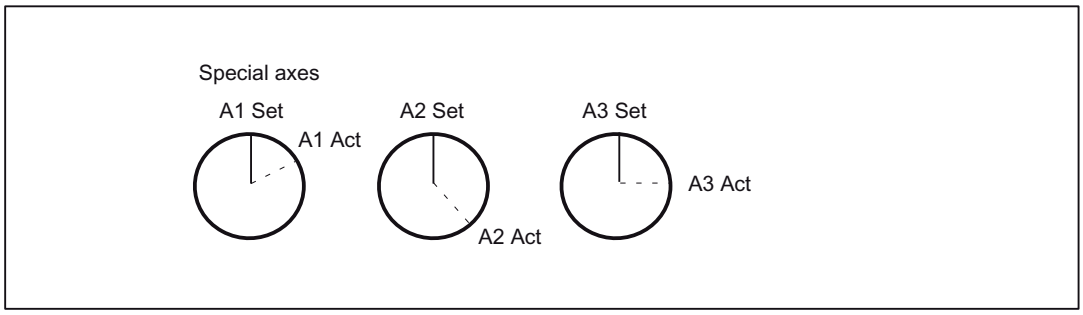


Figure 8-14 Preset actual value memory for special axes only

The values of the following variables are evaluated for measurement type 15:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Actual values of the axes
$AA_MEAS_SETPOINT[axis]	Setpoint position of the individual axes *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_TYPE	15

* optional

The following output variables are written for measurement type 15:

Output variable	Description
$AC_MEAS_FRAME	Result frame with translations

8.4.3.8 Measurement of an oblique edge (measurement type 16)

Measurement of an oblique edge ($AC_MEAS_TYPE = 16)

This measurement determines the position angle of the workpiece and enters it in the frame. A setpoint angle in the +/- 90 degrees range can be input. This can be interpreted as the resultant rotation of the workpiece after the result frame for the active WCS has been activated.

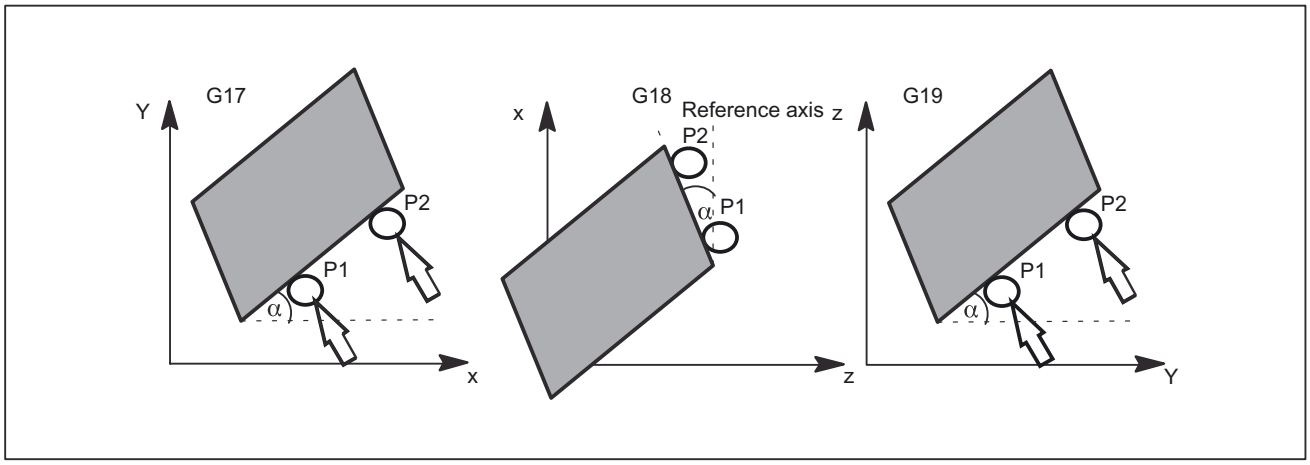


Figure 8-15 Oblique edge in planes G17, G18 and G19

The values of the following variables are evaluated for measurement type 16:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1

Input variable	Description
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_SETANGLE	Setpoint angle *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Unless otherwise specified, the reference coordinate for the alignment of the workpiece is always the abscissa of the selected plane. * =0: Reference coordinate is the abscissa =1: Reference coordinate is the ordinate
$AC_MEAS_INPUT[1]	Unless otherwise specified, the workpiece position angle is entered in the frame as a rotation. Otherwise, a channel axis index can be specified for a rotary axis whose translation is set to the current rotary axis position plus the calculated rotation. The workpiece is then aligned at rotary axis position = 0. The current rotary axis value must be set in $AA_MEAS_POINT[axis]. *
$AC_MEAS_TYPE	16

* optional

The following output variables are written for measurement type 16:

Output variable	Description
$AC_MEAS_FRAME	Result frame with rotation
$AC_MEAS_WP_ANGLE	Calculated workpiece position angle

8.4.3.9 Measurement of an oblique angle in a plane (measurement type 17)

Measurement of an angle in a plane ($AC_MEAS_TYPE = 17)

The oblique plane is determined using three measuring points P1, P2 and P3.

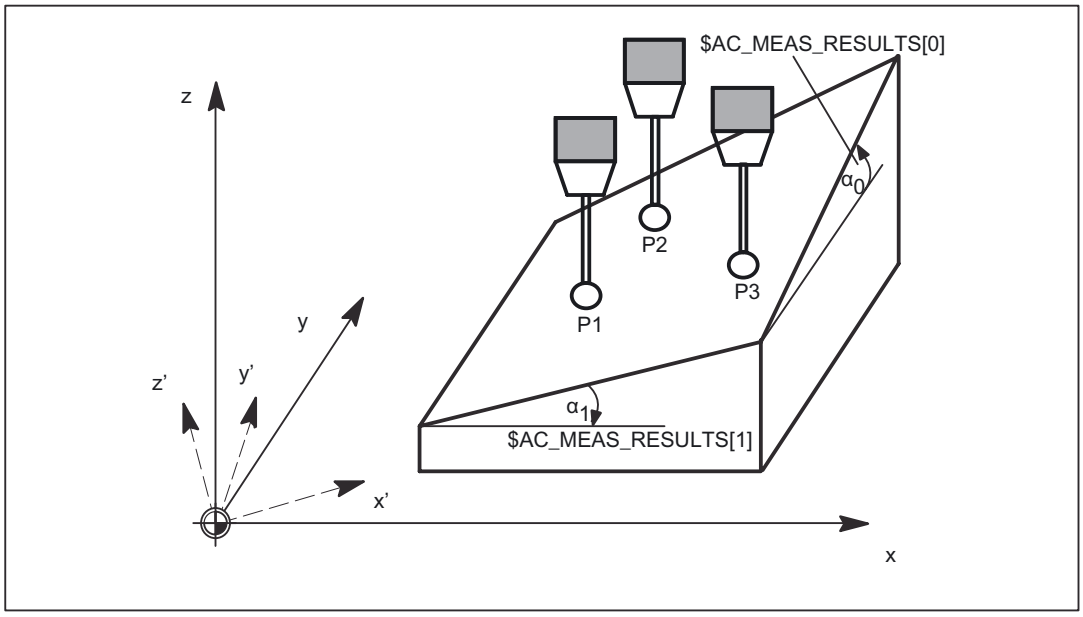


Figure 8-16 Oblique plane in G17

$AC_MEAS_TYPE = 17 defines two resulting angles α_0 and α_1 for the skew of the plane; these are entered in $AC_MEAS_RESULTS[0..1]:

- $AC_MEAS_RESULTS[0] → Rotation at the abscissa
- $AC_MEAS_RESULTS[1] → Rotation at the ordinate

These angles are calculated by means of the three measuring points P1, P2 and P3. In this type of measurement the angle for the applicate ($AC_MEAS_RESULTS[2]) is always pre-filled with 0.

A setpoint rotation that is entered in the result frame can be input for the abscissa and/or the ordinate. If only one angle is specified with a setpoint, the second angle is calculated such that the three measuring points are on an oblique plane with the setpoint angle. Only rotations are entered in the result frame, the WCS reference point is retained. The WCS is rotated such that z' is perpendicular to the oblique plane.

The following plane settings are defined for measurement type 17:

Axis identifier	G17	G18	G19
Abscissa	x axis	z axis	y axis
Ordinate	y axis	x axis	z axis
Applicate (infeed axis)	z axis	y axis	x axis

The values of the following variables are evaluated for measurement type 17:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_SETANGLE[axis]	Setpoint rotations around abscissa and ordinate *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Unless otherwise specified, the points are not projected in a plane * 0: Points are not projected on a plane 1: Points are projected in the active plane or in the selected plane
$AC_MEAS_TYPE	17

* optional

The following output variables are written for measurement type 17:

Output variable	Description
$AC_MEAS_FRAME	Result frame
$AC_MEAS_RESULTS[0]	Angles around abscissa from which three measuring points are calculated
$AC_MEAS_RESULTS[1]	Angles around ordinate from which three measuring points are calculated
$AC_MEAS_RESULTS[2]	Angles around applicate from which three measuring points are calculated
$AC_MEAS_RESULTS[3]	Angle around abscissa which is entered in the result frame
$AC_MEAS_RESULTS[4]	Angle around ordinate which is entered in the result frame
$AC_MEAS_RESULTS[5]	Angle around applicate which is entered in the result frame

Example

Measure angle of a plane

```
DEF INT RETVAL
DEF AXIS _XX, _YY, _ZZ

T1 D1                                   ; Activate probe
```

```
G54                                    ; Activate all frames and G54

$AC_MEAS_VALID = 0                     ; Set all input values to invalid

$AC_MEAS_TYPE = 17                     ; Set measurement type for oblique plane
$AC_MEAS_ACT_PLANE = 0                 ; Measuring plane is G17

_XX=$P_AXN1                            ; Define axes according to the plane
_YY=$P_AXN2
_ZZ=$P_AXN3

G17 G1 _XX=10 _YY=10 F1000             ; Approach 1st measuring point
MEAS = 1 _ZZ=...
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]     ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]     ; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]     ; Assign measurement value to applicate

G1 _XX=20 _YY=10 F1000                 ; Approach 2nd measuring point
MEAS = 1 _ZZ=...
$AA_MEAS_POINT2[_xx] = $AA_MW[_xx]     ; Assign measurement value to abscissa
$AA_MEAS_POINT2[_yy] = $AA_MW[_yy]     ; Assign measurement value to ordinate
$AA_MEAS_POINT2[_zz] = $AA_MW[_zz]     ; Assign measurement value to applicate

G1 _XX=20 _YY=20 F1000                 ; Approach 3rd measuring point
MEAS = 1 _ZZ=...
$AA_MEAS_POINT3[_xx] = $AA_MW[_xx]     ; Assign measurement value to abscissa
$AA_MEAS_POINT3[_yy] = $AA_MW[_yy]     ; Assign measurement value to ordinate
$AA_MEAS_POINT3[_zz] = $AA_MW[_zz]     ; Assign measurement value to applicate

                                       ; Define setpoints for angle
$AA_MEAS_SETANGLE[_xx] = 12            ; Rotation around the abscissa
$AA_MEAS_SETANGLE[_yy] = 4             ; Rotation around the ordinate

$AC_MEAS_FRAME_SELECT = 102            ; Select target frame - G55

$AC_MEAS_T_NUMBER = 1                  ; Select tool
$AC_MEAS_D_NUMBER = 1

RETVAL = MEASURE()                     ; Start measurement calculation

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AC_MEAS_RESULTS[0] <> 12
```

```
setal(61000 + $AC_MEAS_RESULTS[0])
endif

if $AC_MEAS_RESULTS[1] <> 4
setal(61000 + $AC_MEAS_RESULTS[1])
endif

$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame in data management
                                          (G55)

G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

8.4.3.10 Redefine measurement around a WCS reference frame (measurement type 18)

Redefine WCS coordinate system ($AC_MEAS_TYPE = 18)

The zero point of the new WCS is determined by measuring point P1 at surface normal on the oblique plane.

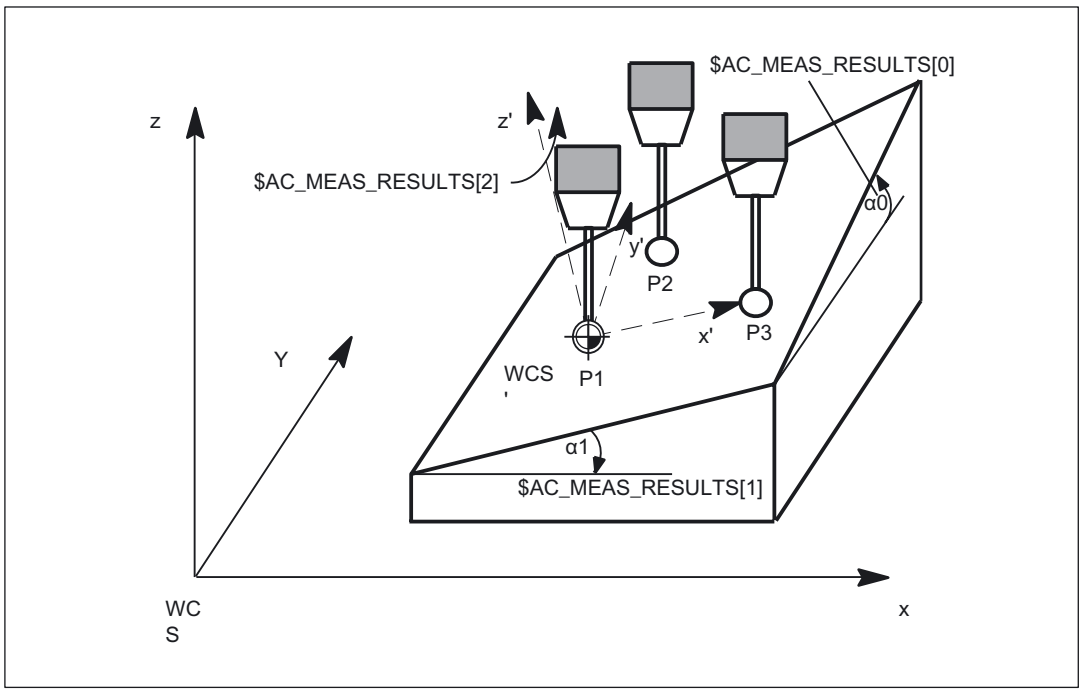


Figure 8-17 Oblique plane in G17

Measurement of plane

The plane is measured in one measuring cycle. The cycle records the three measuring points and passes the necessary values to the variable interface.

The MEASURE() function calculates the solid angles and translational offset of the new WCS' on the basis of the input values.

Transformation of measuring frame

The results of the calculation, i.e. the solid angles and translation, are entered in measuring frame $AC_MEAS_FRAME. The measuring frame is transformed according to the selected frame in the frame chain. The solid angles are also stored in the output values $AC_MEAS_RESULTS[0..2]. In

- The angle around the abscissa of the old WCS is stored in $AC_MEAS_RESULTS[0],
- The angle around the ordinate is stored in $AC_MEAS_RESULTS[1] and
- The angle around the applicate is stored in $AC_MEAS_RESULTS[2].

Define the new WCS' zero

After performing the calculation, the measuring cycle can write and activate the selected frame in the frame chain with the measuring frame. After activation, the new WCS is positioned at surface normal on the inclined plane, with measuring point P1 as the zero point of the new WCS.

The programmed positions then refer to the inclined plane.

Application

CAD systems often define inclined planes by specifying three points P1, P2 and P3 on this plane. In this case,

- 1st measuring point P1 is applied as the new WCS' reference point,
- 2nd measuring point P2 defines the direction of the abscissa x' of the newly rotated WCS' coordinate system and the
- 3rd measuring point P3 is used to determine the solid angles.

The values of the following variables are evaluated for measurement type 18:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_SETPOINT[axis]	Setpoint position of P1 *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_INPUT[0]	Unless otherwise specified, the points are not projected in a plane * 0: Points are not projected on a plane 1: Points are projected in the active plane or in the selected plane
$AC_MEAS_TYPE	18

* optional

The following output variables are written for measurement type 18:

Output variable	Description
$AC_MEAS_FRAME	Result frame with rotations and transformation
$AC_MEAS_RESULTS[0]	Calculated angle around the abscissa
$AC_MEAS_RESULTS[1]	Calculated angle around the ordinate
$AC_MEAS_RESULTS[2]	Calculated angle around the applicate

Example

Workpiece coordinate system on the inclined plane

```
DEF INT RETVAL
DEF AXIS _XX, _YY, _ZZ

T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54

$AC_MEAS_VALID = 0                      ; Set all input values to invalid

$AC_MEAS_TYPE = 18                      ; Set measurement type for oblique plane
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17

_XX=$P_AXN1                             ; Define axes according to the plane
_YY=$P_AXN2
_ZZ=$P_AXN3

G17 G1 _XX=10 _YY=10 F1000              ; Approach 1st measuring point
MEAS = 1 _ZZ=...

$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measurement value to applicate

G1 _XX=20 _YY=10 F1000                  ; Approach 2nd measuring point
MEAS = 1 _ZZ=...

$AA_MEAS_POINT2[_xx] = $AA_MW[_xx]      ; Assign measurement value to abscissa
$AA_MEAS_POINT2[_yy] = $AA_MW[_yy]      ; Assign measurement value to ordinate
$AA_MEAS_POINT2[_zz] = $AA_MW[_zz]      ; Assign measurement value to applicate

G1 _XX=20 _YY=20 F1000                  ; Approach 3rd measuring point
MEAS = 1 _ZZ=...

$AA_MEAS_POINT3[_xx] = $AA_MW[_xx]      ; Assign measurement value to abscissa
```

```
$AA_MEAS_POINT3[_yy] = $AA_MW[_yy]      ; Assign measurement value to ordinate
$AA_MEAS_POINT3[_zz] = $AA_MW[_zz]      ; Assign measurement value to applicate

$AA_MEAS_SETPOINT[_xx] = 10             ; Define setpoints for P1
$AA_MEAS_SETPOINT[_yy] = 10
$AA_MEAS_SETPOINT[_zz] = 10

$AC_MEAS_FRAME_SELECT = 102             ; Select target frame - G55

$AC_MEAS_T_NUMBER = 1                   ; Select tool
$AC_MEAS_D_NUMBER = 1

RETVAL = MEASURE()                      ; Start measurement calculation

if RETVAL <> 0
setal(61000 + RETVAL)
endif

                                        ; Calculation results for the solid angles
                                        ; Angle around the
R0 = $AC_MEAS_RESULTS[0]                ; Abscissa for the old WCS
R1 = $AC_MEAS_RESULTS[1]                ; Ordinate
R2 = $AC_MEAS_RESULTS[2]                ; Applicate

$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame in data management
                                          (G55)

G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

8.4.3.11 Measurement of a 1-, 2- and 3-dimensional setpoint selection (measurement type 19, 20, 21)

1-dimensional setpoint value ($AC_MEAS_TYPE = 19)

With this measurement method, it is possible to define exactly one setpoint for the abscissa, the ordinate and the applicate. If two or three setpoints are defined, only the first is accepted in the sequence abscissa, ordinate and applicate. The tool is not taken into account.

It is purely an actual value memory preset for the abscissa, the ordinate or the applicate.

The values of the following variables are evaluated for measurement type 19:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for the abscissa
$AA_MEAS_POINT1[axis]	Measuring point 1 for the ordinate

Input variable	Description
$AA_MEAS_POINT1[axis]	Measuring point 1 for the applicate
$AA_MEAS_SETPOINT[axis]	Setpoint position of abscissa or ordinate or applicate
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_FINE_TRANS	Unless otherwise specified, frame is written to coarse translation *
$AC_MEAS_TYPE	19

* optional

The following output variables are written for measurement type 19:

Output variable	Description
$AC_MEAS_FRAME	Result frame with rotations and translation

Example

1-dimensional setpoint selection

```
DEF INT RETVAL
DEF REAL _CORMW_XX,
_CORMW_YY,
_CORMW_ZZ
DEF AXIS _XX, _YY, _ZZ
T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54
$AC_MEAS_VALID = 0                      ; Set all input values to invalid
$AC_MEAS_TYPE = 19                      ; Set measurement type for 1-dimensional
                                          setpoint selection
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17
_XX=$P_AXN1                             ; Define axes according to the plane
_YY=$P_AXN2
_ZZ=$P_AXN3
                                        ; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measurement value to applicate
$AA_MEAS_SETPOINT[_xx] = 10             ; Define setpoint for abscissa
$AC_MEAS_FRAME_SELECT = 102             ; Select target frame - G55
RETVAL = MEASURE()                      ; Start measurement calculation
if RETVAL <> 0
setal(61000 + RETVAL)
endif
$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame in data management
                                          (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

2-dimensional setpoint value ($AC_MEAS_TYPE = 20)

Setpoints for two dimensions can be defined using this measuring method. Any combination of 2 out of 3 axes is permissible. If three setpoints are specified, only the values for the abscissa and the ordinate are accepted. The tool is not taken into account.

This is a purely actual value memory preset.

The values of the following variables are evaluated for measurement type 20:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for the abscissa
$AA_MEAS_POINT1[axis]	Measuring point 1 for the ordinate
$AA_MEAS_POINT1[axis]	Measuring point 1 for the applicate
$AA_MEAS_SETPOINT[axis]	Setpoint position for the 1st dimension
$AA_MEAS_SETPOINT[axis]	Setpoint position for the 2nd dimension
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_FINE_TRANS	Unless otherwise specified, frame is written to coarse translation *
$AC_MEAS_TYPE	20

* optional

The following output variables are written for measurement type 20:

Output variable	Description
$AC_MEAS_FRAME	Result frame with rotations and translation

Example

2-dimensional setpoint selection

```
DEF INT RETVAL
DEF REAL _CORMW_XX,
_CORMW_YY,
_CORMW_ZZ
DEF AXIS _XX, _YY, _ZZ
T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54
$AC_MEAS_VALID = 0                      ; Set all input values to invalid
$AC_MEAS_TYPE = 20                      ; Set measurement type for 2-dimensional
                                          setpoint selection
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17
_XX=$P_AXN1                             ; Define axes according to the plane
_YY=$P_AXN2
_ZZ=$P_AXN3
                                        ; Assign measured values
```

```
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measurement value to applicate
$AA_MEAS_SETPOINT[_xx] = 10             ; Define setpoint for abscissa and ordinate
$AA_MEAS_SETPOINT[_yy] = 10
$AC_MEAS_FRAME_SELECT = 102             ; Select target frame - G55
RETVAL = MEASURE()                      ; Start measurement calculation
if RETVAL <> 0
setal(61000 + RETVAL)
endif
$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame in data management
                                          (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

3-dimensional setpoint value ($AC_MEAS_TYPE = 21)

Using this measurement method, it is possible to define a setpoint for the abscissa, the ordinate and the applicate. The tool is not taken into account.

It is purely an actual value memory preset for the abscissa, ordinate and applicate.

The values of the following variables are evaluated for measurement type 21:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for the abscissa
$AA_MEAS_POINT1[axis]	Measuring point 1 for the ordinate
$AA_MEAS_POINT1[axis]	Measuring point 1 for the applicate
$AA_MEAS_SETPOINT[axis]	Setpoint position for the abscissa
$AA_MEAS_SETPOINT[axis]	Setpoint position for the ordinate
$AA_MEAS_SETPOINT[axis]	Setpoint position for the applicate
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_FINE_TRANS	Unless otherwise specified, frame is written to coarse translation *
$AC_MEAS_TYPE	21

* optional

The following output variables are written for measurement type 21:

Output variable	Description
$AC_MEAS_FRAME	Result frame with rotations and translation

Example

3-dimensional setpoint selection

```
DEF INT RETVAL
DEF REAL _CORMW_XX,
_CORMW_YY,
_CORMW_ZZ
DEF AXIS _XX, _YY, _ZZ
T1 D1                                   ; Activate probe
G54                                     ; Activate all frames and G54
$AC_MEAS_VALID = 0                      ; Set all input values to invalid
$AC_MEAS_TYPE = 21                      ; Set measurement type for 3-dimensional
                                          setpoint selection
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17
_XX=$P_AXN1                             ; Define axes according to the plane
_YY=$P_AXN2
_ZZ=$P_AXN3
                                        ; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_MW[_xx]      ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_MW[_yy]      ; Assign measurement value to ordinate
$AA_MEAS_POINT1[_zz] = $AA_MW[_zz]      ; Assign measurement value to applicate
$AA_MEAS_SETPOINT[_xx] = 10             ; Define setpoint for abscissa, ordinate and
                                          applicate
$AA_MEAS_SETPOINT[_yy] = 10             ; Define
$AA_MEAS_SETPOINT[_zz] = 10
$AC_MEAS_FRAME_SELECT = 102             ; Select target frame - G55
$AA_MEAS_SETPOINT[_yy] = 10
RETVAL = MEASURE()                      ; Start measurement calculation
if RETVAL <> 0
setal(61000 + RETVAL)
endif
$P_UIFR[2] = $AC_MEAS_FRAME             ; Write measurement frame in data management
                                          (G55)
G55 G0 AX[_xx]=10 AX[_yy]=10            ; Activate frame and traverse
m30
```

8.4.3.12 Measurement of an oblique angle (measurement type 24)

Measurement method for converting a measuring point in any coordinate system

Coordinate transformation of a position ($AC_MEAS_TYPE = 24)

With this method of measurement, a measuring point in any coordinate system (WCS, BCS, MCS) can be converted with reference to a new coordinate system by coordinate transformation.

The new coordinate system is generated by specifying a desired frame chain.

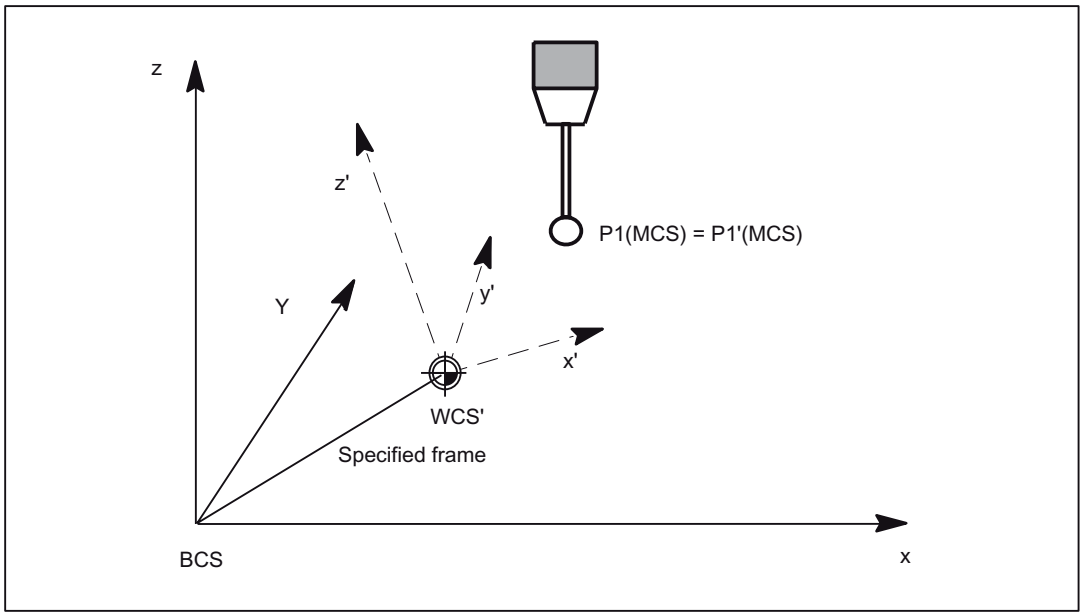


Figure 8-18 Coordinate transformation of a position

The values of the following variables are evaluated for measurement type 24:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Position to be transformed
$AC_MEAS_P1:COORD	Default is 0: WCS, 1: BCS, 2: MCS *
$AC_MEAS_P2_COORD	Target coordinate system *
$AC_MEAS_TOOL_MASK	0x20; Length of the active tool is included in the coordinate transformation of a position *
$AC_MEAS_CHSFR	System frames from data management *
$AC_MEAS_NCBFR	Global basic frames from the data management *
$AC_MEAS_CHBFR	Channel basic frames from the data management *
$AC_MEAS_UIFR	Settable frame from data management *
$AC_MEAS_PFRAME	1: Programmable frame is not included in calculation *
$AC_MEAS_TYPE	24

* optional

The following output variables are written for measurement type 24:

Output variable	Description
$AC_MEAS_POINT2[axis]	Converted axis positions

Example

WCS coordinate transformation of a measured position

```
DEF INT RETVAL
DEF INT LAUF
DEF REAL _CORMW_xx, _CORMW_yy, _CORMW_zz
DEF AXIS _XX, _YY, _ZZ

$TC_DP1[1,1]=120                          ; Tool type end mill
$TC_DP2[1,1]=20
$TC_DP3[1,1]=0                            ; (z) length compensation vector
$TC_DP4[1,1]=0                            ; (y) length compensation vector
$TC_DP5[1,1]=0                            ; (x) length compensation vector
$TC_DP6[1,1]=2                            ; Radius

T1 D1                                     ; Activate probe

G17                                       ; Oblique plane G17

_xx=$P_AXN1 _yy=$P_AXN2 _zz=$P_AXN3       ; Define axes according to the plane

                                          ; Entire frame results in
                                            CTRANS(_xx,10,_yy,-1,_zz,5,A,6,B,7)
$P_CHBFR[0]=CTRANS(_zz,5,A,6) : CROT(_zz,45)
$P_UIFR[1]=CTRANS( )
$P_UIFR[1,_xx,TR]=-SIN(45)
$P_UIFR[1,_yy,TR]=-SIN(45)
$P_UIFR[2]=CTRANS( )
$P_PFRAME=CROT(_zz,-45)
$P_CYCFR=CTRANS(_xx,10,B,7)

G54                                       ; Activate all frames and G54

G0 X0 Y0 Z0 A0 B0 F1000

$AC_MEAS_VALID = 0                        ; Set all input values to invalid

$AC_MEAS_TYPE = 24                        ; Set measurement type for coordinate
                                            transformation

$AC_MEAS_ACT_PLANE = 0                    ; Measuring plane is G17

                                          ; Assign measured values
$AA_MEAS_POINT1[_xx] = $AA_IW[_xx]        ; Assign measurement value to abscissa
$AA_MEAS_POINT1[_yy] = $AA_IW[_yy]        ; Assign measurement value to ordinate
```

```
$AA_MEAS_POINT1[_zz] = $AA_IW[_zz]      ; Assign measurement value to applicate
$AA_MEAS_POINT1[A] = $AA_IW[A]
$AA_MEAS_POINT1[B] = $AA_IW[B]

$AC_MEAS_P1_COORD=0                     ; Converting a position from WCS into WCS'
$AC_MEAS_P2_COORD=0
                                        ; Set WCS
                                        ; Entire frame results in
                                          CTRANS(_xx,0,_yy,0,_zz,5,A,6,B,0)

                                        ; Stop cycle frame
$AC_MEAS_CHSER=$MC_MM_SYSTEM_FRAME_MASK B_AND 'B1011111'

$AC_MEAS__NCBFR='B0'                    ; Stop global basic frame
$AC_MEAS__CHBFR='B1'                    ; Channel basic frame 1 from data management
$AC_MEAS__UIFR=2                        ; Settable frame G55 from data management

$AA_MEAS_PFRAME=1                       ; Do not include programmable frame in
                                          calculation

RETVAL = MEASURE()                      ; Start measurement calculation

if RETVAL <> 0
setal(61000 + RETVAL)
endif

if $AA_MEAS_PIONT2[_xx] <> 10
setal(61000)
M0
stopre
endif
if $AA_MEAS_PIONT2[_yy] <> -1
setal(61000)
M0
stopre
if $AA_MEAS_PIONT2[_zz] <> 0
setal(61000)
M0
stopre
if $AA_MEAS_PIONT2[A] <> 0
setal(61000)
M0
stopre
if $AA_MEAS_PIONT2[B] <> 7
setal(61000)
M0
stopre

m30
```

8.4.3.13 Measurement of a rectangle (measurement type 25)

Measuring points for determining a rectangle ($AC_MEAS_TYPE = 25)

To determine a rectangle, tool dimensions are required in the following working planes.

- G17 working plane x/y infeed direction z
- G18 working plane z/x infeed direction y
- G19 working plane y/z infeed direction x

Four measuring points are required per rectangle.

Measuring points can be specified in any desired order. The measuring points with the largest ordinate distance correspond to points P3 and P4.

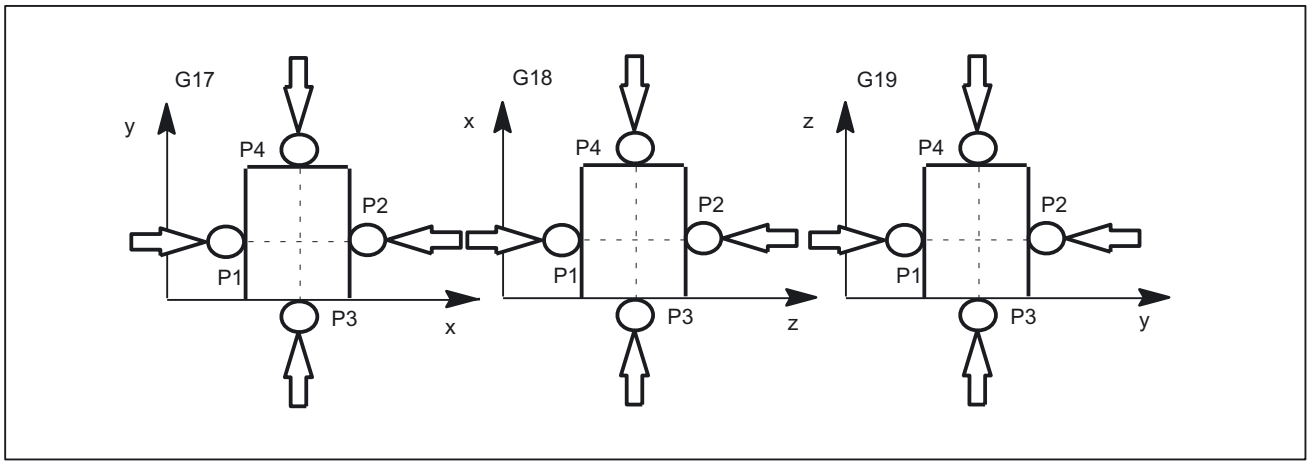


Figure 8-19 Determining a rectangle with infeed into the working plane G17, G18 and G19

The values of the following variables are evaluated for measurement type 25:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_POINT2[axis]	Measuring point 2
$AA_MEAS_POINT3[axis]	Measuring point 3
$AA_MEAS_POINT4[axis]	Measuring point 4
$AA_MEAS_SETPOINT[axis]	Setpoint position of web center *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FINE_TRANS	0: Coarse offset, 1: Fine offset *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *

Input variable	Meaning
$AC_MEAS_INPUT[0]	Without specification of outer corner * =0: Measurement for outer corner =1: Measurement for inner corner
$AC_MEAS_TYPE	25

* optional

The following output variables are written for measurement type 25:

Output variable	Meaning
$AC_MEAS_FRAME	Result frame with translation
$AC_MEAS_RESULTS[0]	Abscissa of the calculated center point
$AC_MEAS_RESULTS[1]	Ordinate of the calculated center point
$AC_MEAS_RESULTS[2]	Applicate of the calculated center point
$AC_MEAS_RESULTS[3]	Width of rectangle P1/P2
$AC_MEAS_RESULTS[4]	Length of rectangle P3/P4

8.4.3.14 Measurement for saving data management frames (measurement type 26)

Saving data management frames ($AC_MEAS_TYPE = 26)

This measurement type offers the option of saving some or all data management frames with their current value assignments to a file. The measurement can be initiated by executing a command or via the part program. The function can also be activated from different channels. The files are set up in directory _N_SYF_DIR.

A Restore operation deletes the backed-up data and a new Save operation overwrites the existing back-up. The data last saved can then be deleted with

- $AC_MEAS_CHSFR = 0 system frames;
- $AC_MEAS_NCBFR = 0 global basic frames;
- $AC_MEAS_CHBFR = 0 channel basic frames;
- $AC_MEAS_UIFR = 0 number of settable frames

from the data management system using a second Save operation.

Note

If you decide to create a backup of all data management frames, remember that 1 KB of memory is needed to save each frame. If insufficient memory is available, the process is aborted with error message MEAS_NO_MEMORY. The required amount of DRAM can be modified using the following machine data:

MD18351 $MM_DRAM_FILE_MEM_SIZE

The values of the following variables are evaluated for measurement type 26:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AC_MEAS_CHSFR	Bit mask system frames from data management. * If this variable is not written, all system frames are backed up.
$AC_MEAS_NCBFR	Bit mask of global basic frames from the data management. * If this variable is not written, all global basic frames are backed up.
$AC_MEAS_CHBFR	Bit mask of channel basic frames from the data management. * If this variable is not written, all channel basic frames are backed up.
$AC_MEAS_UIFR	Number of settable frames from data management. * 0..100: 1: G500 2: G500, G54. If this variable is not written, all settable frames are backed up.
$AC_MEAS_TYPE	26

8.4.3.15 Measurement for restoring backed-up data management frames (measurement type 27)

Restoring data management frames last backed up ($AC_MEAS_TYPE = 27)

This measurement type allows data management frames backed up by measurement type 26 to be restored to the SRAM.

It is possible to restore either some or all of the frames last backed up. If a frame that has not been backed up is selected, the selection is ignored. The process is not aborted.

The values of the following variables are evaluated for measurement type 27:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AC_MEAS_CHSFR	Bit mask system frames from data management. * If this variable is not written, all system frames are restored.
$AC_MEAS_NCBFR	Bit mask of global basic frames from the data management. * If this variable is not written, all global basic frames are restored.
$AC_MEAS_CHBFR	Bit mask of channel basic frames from the data management. * If this variable is not written, all channel basic frames are restored.
$AC_MEAS_UIFR	Number of settable frames from data management. * Range of 1: G54 to G99: G599. If this variable is not written, all settable frames are restored.
$AC_MEAS_TYPE	27

* optional

8.4.3.16 Measurement for defining an additive rotation for taper turning (measurement type 28)

Taper turning

Additive rotation of plane ($AC_MEAS_TYPE = 28)

This measurement type 28 is used via the ManualTurn Advanced user interface for the taper turning application. An additive rotation of the active (or a certain) plane around an angle in the range of α = +/- 90° can be specified with it. The rotation takes place on the coordinate axis at right angles to the plane.

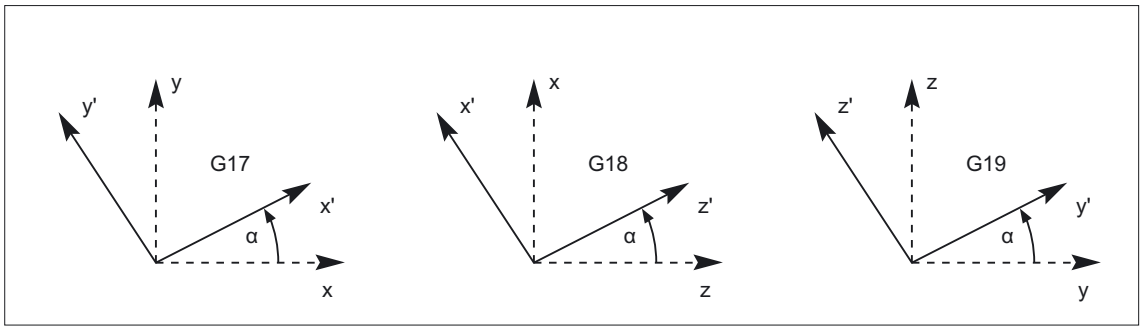


Figure 8-20 Rotation of the planes G17, G18 and G19 by angle α = +30°

Application

With taper turning, the active plane is rotated by the taper angle, whereby the rotation is written in the active cycle frame. With RESET, the cycle frame is deleted. Re-activation may be necessary. The selection of the cycle frame is made depending on the SZS position display. If after activation of the rotation, e.g. with active plane G18, traversing is performed in the direction of z', the actual positions of the corresponding axes change **simultaneously** for **x and z**

Rotations with active planes G17 and G18 behave in the same way and are displayed in the above figure. The values of the following variables are evaluated for measurement type 28:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AC_MEAS_WP_SETANGLE	Setpoint angle
$AC_MEAS_ACT_PLANE	Rotation is through the active plane unless otherwise specified. *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified. *
$AC_MEAS_INPUT[0]	1: Taper turning is active. *
$AC_MEAS_TYPE	28

* optional

The following output variables are written for measurement type 28:

Output variable	Description
$AC_MEAS_FRAME	Result with rotation

8.4.4 Tool measuring

The control calculates the distance between the tool tip and the tool carrier reference point T from the tool length specified by the user.

The following measurement types can be used to measure a tool loaded on a turning or milling machine:

Measurement types	Tool measuring
$AC_MEAS_TYPE = 10	Tool lengths on a reference part that has already been measured
$AC_MEAS_TYPE = 11	Tool diameter on a reference part that has already been measured
$AC_MEAS_TYPE = 22	Tool diameters on machines with zoom-in function (ShopTurn)
$AC_MEAS_TYPE = 23	Tool lengths with stored or current positions (ShopTurn) Measurement of a tool length of two tools with the following orientation: Two turning tools with: Their own reference point each for tool orientation in the approach direction. One reference point for tool position that is opposite to the approach direction and tool orientation. Two milling tools with: Their own reference point each for tool orientation in -y. One reference point for tool position in -y and a tool position opposite to the approach direction. Two milling tools rotated 90 degrees with: Their own reference point each for tool orientation in the approach direction. One reference point for a tool position that is opposite to the approach direction and tool orientation.

8.4.5 Types of workpiece measurement

8.4.5.1 Measurement of tool lengths (measurement type 10)

Tool length measurement on a reference part that has already been measured ($AC_MEAS_TYPE = 10)

The tool length can be measured on a reference part that has already been measured. Depending on the position of the tool, it is possible to select plane G17 for tool position in the z direction, G18 for tool position in the y direction and G19 for tool position in the x direction.

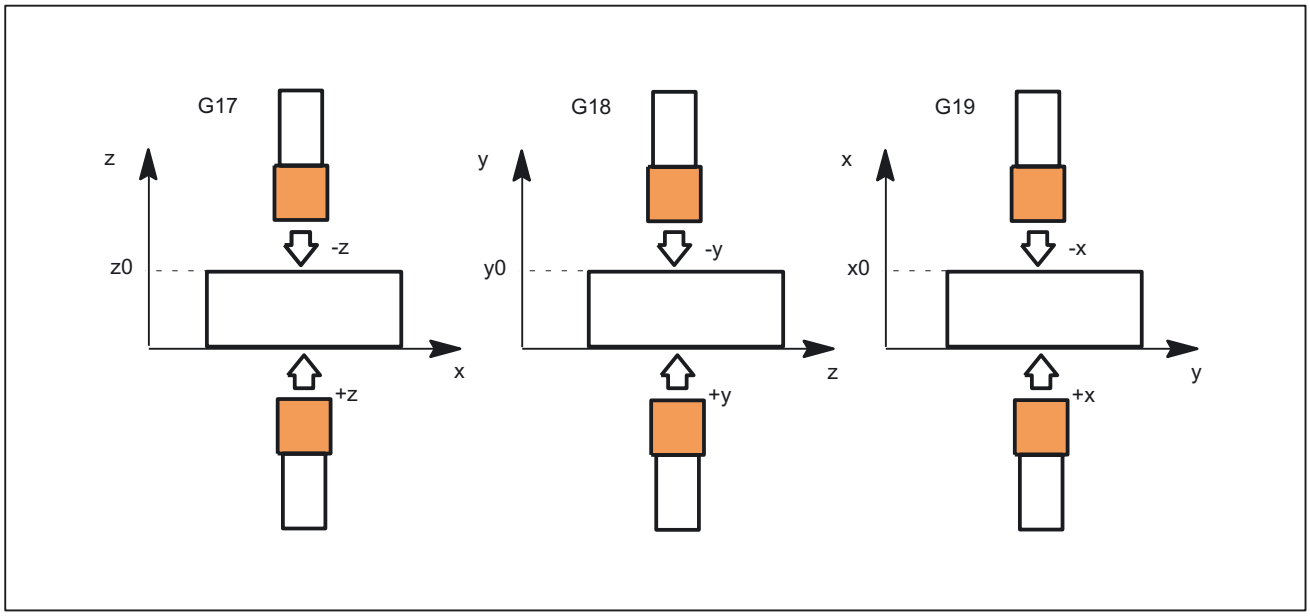


Figure 8-21 Tool length measurement for the selected plane G17, G18 and G19

The values of the following variables are evaluated for measurement type 10:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AC_MEAS_P1_COORD	Coordinate system of the measuring point *
$AA_MEAS_SETPOINT[axis]	Set position z0
$AC_MEAS_SET_COORD	Coordinate system of setpoint *
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_TYPE	10

* optional

The following output variables are written for measurement type 10:

Output variable	Description
$AC_MEAS_TOOL_LENGTH	Tool length
$AC_MEAS_RESULTS[0]	Tool length in x
$AC_MEAS_RESULTS[1]	Tool length in y
$AC_MEAS_RESULTS[2]	Tool length in z
$AC_MEAS_RESULTS[3]	Tool length L1
$AC_MEAS_RESULTS[4]	Tool length L2
$AC_MEAS_RESULTS[5]	Tool length L3

Example

Measuring the tool length

```
DEF INT RETVAL
T0 D0
g0 x0 y0 z0 f10000
                                        ; Measure tool length
$AC_MEAS_VALID = 0                      ; Set all input values to invalid
g1 z10                                  ; Move tool towards reference part

$AC_MEAS_LATCH[0] = 1                   ; Pick up measuring point 1
$AC_MEAS_DIR_APPROACH = 5               ; Set approach direction -z
$AA_MEAS_SETPOINT[x] = 0                ; Set reference position
$AA_MEAS_SETPOINT[y] = 0
$AA_MEAS_SETPOINT[z] = 0
$AC_MEAS_ACT_PLANE = 0                  ; Measuring plane is G17
$AC_MEAS_T_NUMBER = 0                   ; No tool has been selected
$AC_MEAS_D_NUMBER = 0
$AC_MEAS_TYPE = 10                      ; Set measuring type on tool length

RETVAL = MEASURE()                      , Start measuring process
if RETVAL <> 0 setal(61000 + RETVAL)
endif

if $AC_MEAS_TOOL_LENGTH <> 10           ; Query known tool length
setal(61000 + $AC_MEAS_TOOL_LENGTH)
endif
m30
```

8.4.5.2 Measurement of tool diameter (measurement type 11)

Tool diameter measurement on a reference part ($AC_MEAS_TYPE = 11)

The tool diameter can be measured on a reference part that has already been measured. Depending on the position of the tool, it is possible to select plane G17 for tool position in the z direction, G18 for tool position in the y direction and G19 for tool position in the x direction.

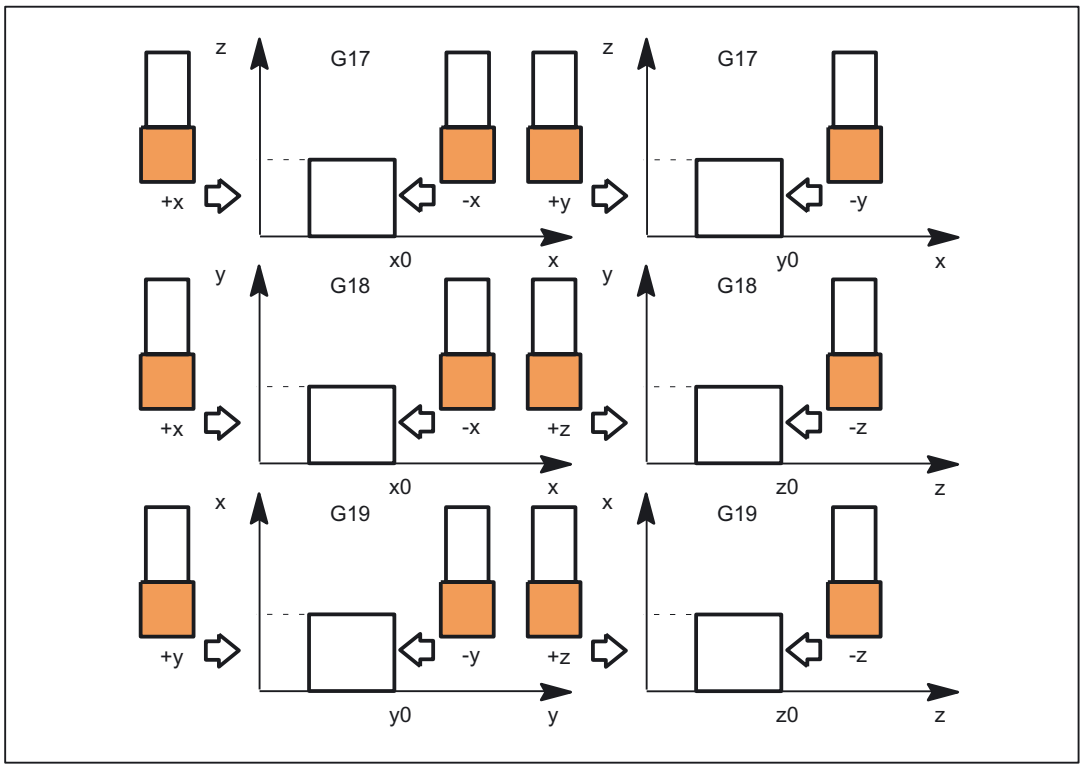


Figure 8-22 Tool diameter for selected planes G17, G18 and G19

The values of the following variables are evaluated for measurement type 11:

Input variable	Meaning
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1
$AA_MEAS_SETPOINT[axis]	Set position x0
$AC_MEAS_DIR_APPROACH	0: +x, 1: -x, 2: +y, 3: -y, 4: +z, 5: -z
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_TYPE	11

* optional

The following output variables are written for measurement type 11:

Output variable	Meaning
$AC_MEAS_TOOL_DIAMETER	Tool diameter

8.4.5.3 Measurement of tool lengths with zoom-in function (measurement type 22)

Tool length with zoom-in function

Tool length measurement with zoom-in function ($AC_MEAS_TYPE = 22)

If a zoom-in function is available on the machine, it can be used to determine the tool dimensions.

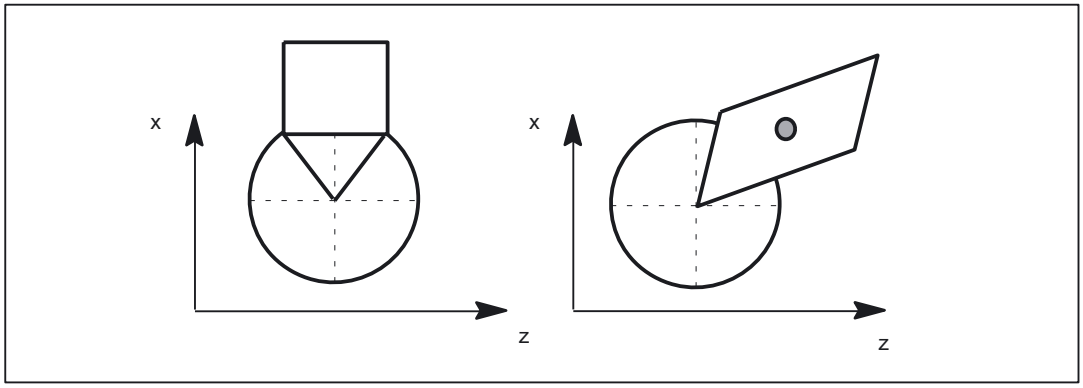


Figure 8-23 Measurement of tool lengths with zoom-in function

The values of the following variables are evaluated for measurement type 22:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Measuring point 1 for all channel axes
$AC_MEAS_P1_COORD	Coordinate system of the measuring point *
$AA_MEAS_SETPOINT[axis]	Zoom positions x and z must be specified
$AC_MEAS_SET_COORD	Coordinate system of setpoint *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_FRAME_SELECT	Calculated as additive frame unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TYPE	22

* optional

The following output variables are written for measurement type 22:

Output variable	Description
$AC_MEAS_RESULT[0]	Tool length in x
$AC_MEAS_RESULT[1]	Tool length in y
$AC_MEAS_RESULT[2]	Tool length in z
$AC_MEAS_RESULT[3]	Tool length L1
$AC_MEAS_RESULT[4]	Tool length L2
$AC_MEAS_RESULT[5]	Tool length L3

8.4.5.4 Measuring a tool length with stored or current position (measurement type 23)

Tool length with stored / current position

Tool length measurement with stored or current position ($AC_MEAS_TYPE = 23)

In the case of manual measurement, the tool dimensions can be determined in the X and Z directions. From the known position of the

- Tool carrier reference point and the
- Workpiece dimensions

ShopTurn calculates the tool offset data.

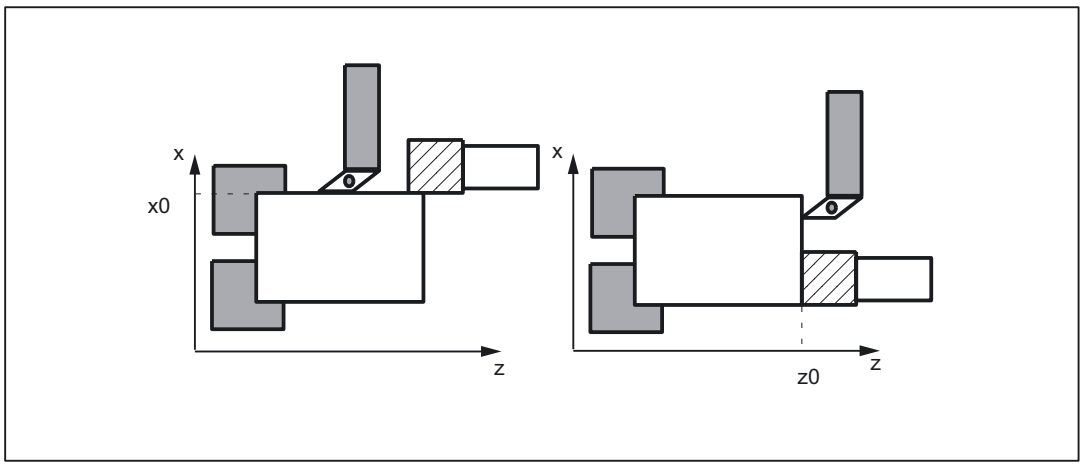


Figure 8-24 Measurement of a tool length with a stored or actual position

The values of the following input variables are evaluated for measurement type 23:

Input variable	Description
$AC_MEAS_VALID	Validity bits for input variables
$AA_MEAS_POINT1[axis]	Current or marked position
$AC_MEAS_P1_COORD	Coordinate system of the measuring point *
$AA_MEAS_SETPOINT[axis]	Setpoint position (minimum one geo axis must be specified)

Input variable	Description
$AC_MEAS_SET_COORD	Coordinate system of setpoint *
$AC_MEAS_ACT_PLANE	Calculated as active plane unless otherwise specified *
$AC_MEAS_T_NUMBER	Calculated as active T unless otherwise specified (T0) *
$AC_MEAS_D_NUMBER	Calculated as active D unless otherwise specified (D0) *
$AC_MEAS_TOOL_MASK	Tool position, radius *
$AC_MEAS_DIR_APPROACH	Approach direction *
$AC_MEAS_INPUT[0] = 1	The calculated tool lengths are written to the data management *
$AC_MEAS_TYPE	23

* optional

The following output variables are written for measurement type 23:

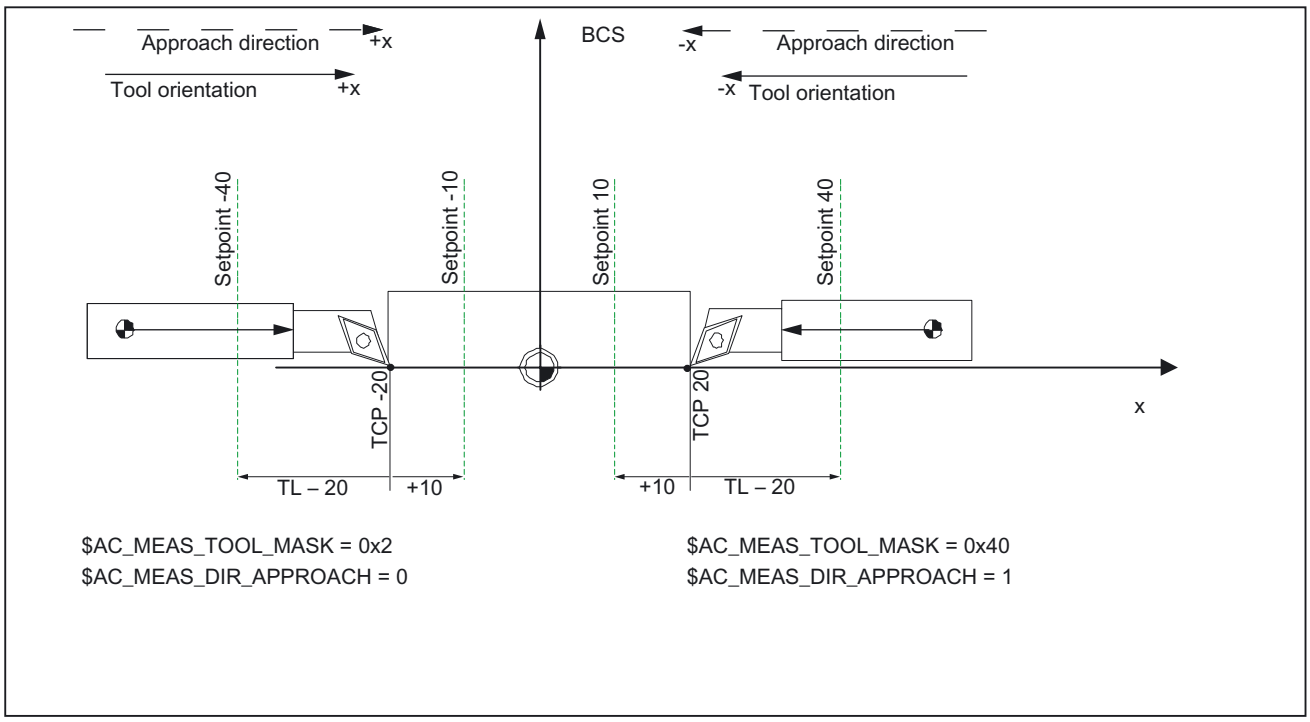
Output variable	Description
$AC_MEAS_RESULT[0]	Tool length in x
$AC_MEAS_RESULT[1]	Tool length in y
$AC_MEAS_RESULT[2]	Tool length in z
$AC_MEAS_RESULT[3]	Tool length L1
$AC_MEAS_RESULT[4]	Tool length L2
$AC_MEAS_RESULT[5]	Tool length L3

8.4.5.5 Measurement of a tool length of two tools with orientation

Tool orientation

Tools oriented towards the tool carrier must be marked by $AC_MEAS_TOOL_MASK = 0x200. The calculated tool lengths are then included negatively.

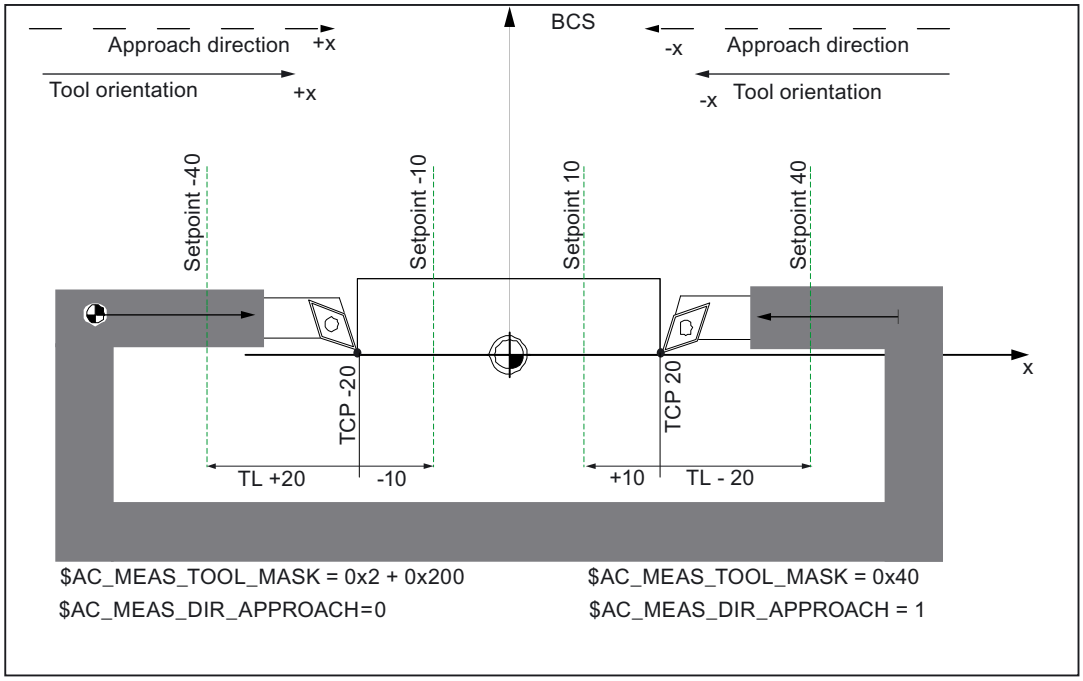
Two turning tools each with their own reference point with a tool orientation in the approach direction



In the case of the tool position of two turning tools each with their own reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

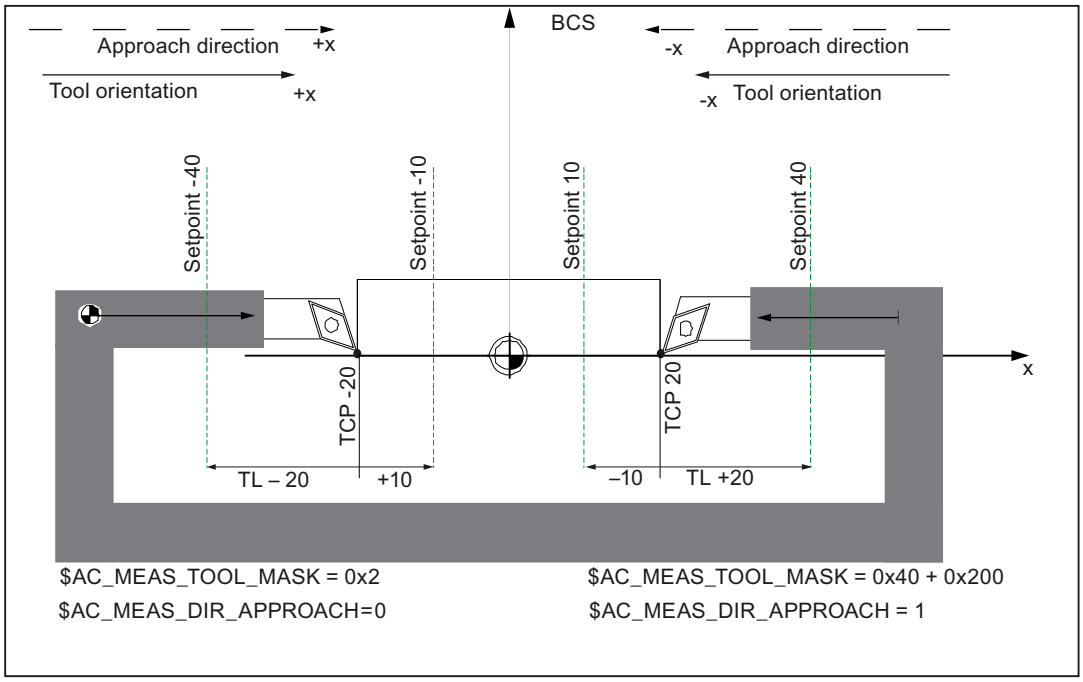
Approach direction and tool orientation +x	Approach direction and tool orientation -x
$AC_MEAS_TOOL_MASK = 0x2	$AC_MEAS_TOOL_MASK = 0x40
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1

Two turning tools with one reference point with a tool position opposite to the orientation



In the case of the tool position of two turning tools with one reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

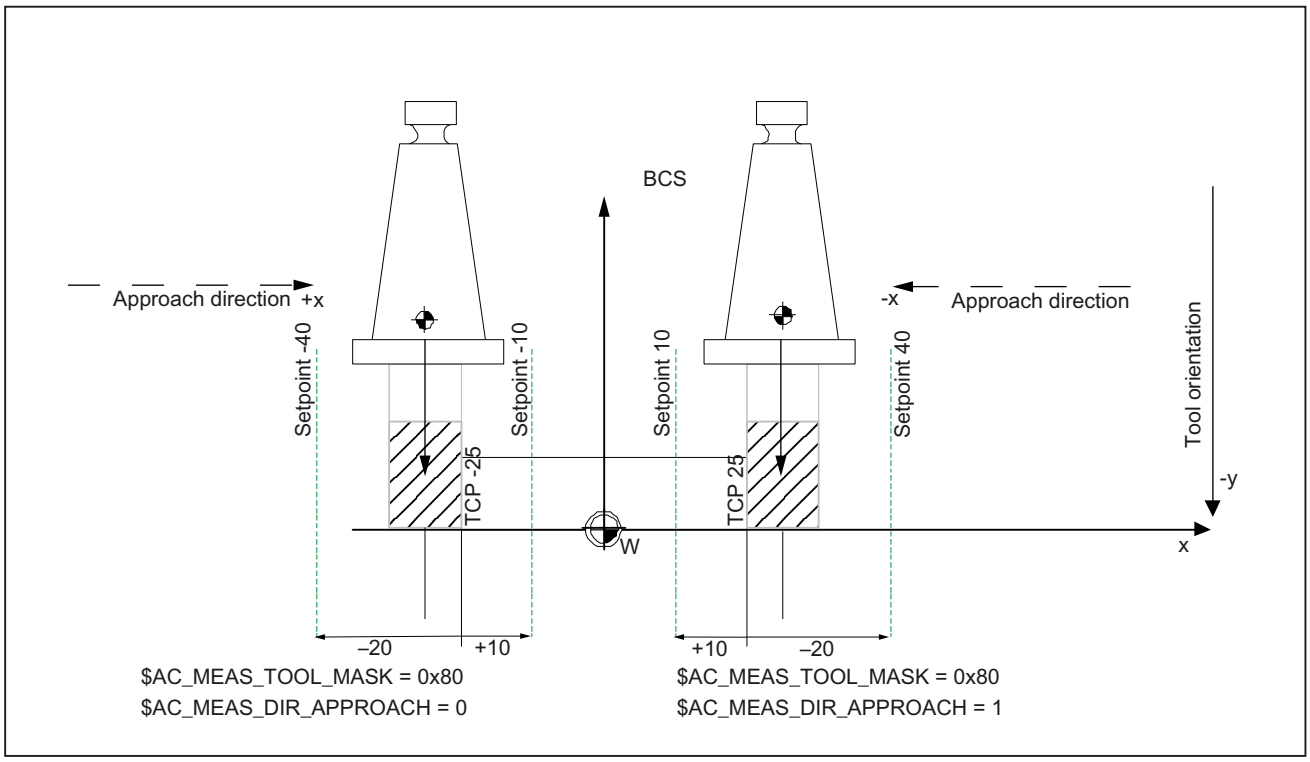
Approach direction and tool orientation +x	Approach direction and tool orientation -x
$AC_MEAS_TOOL_MASK = 0x2 + 0x200	$AC_MEAS_TOOL_MASK = 0x40
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1



In the case of the tool position of two turning tools with one reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

Approach direction and tool orientation +x	Approach direction and tool orientation -x
$AC_MEAS_TOOL_MASK = 0x2	$AC_MEAS_TOOL_MASK = 0x40 + 0x200
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1

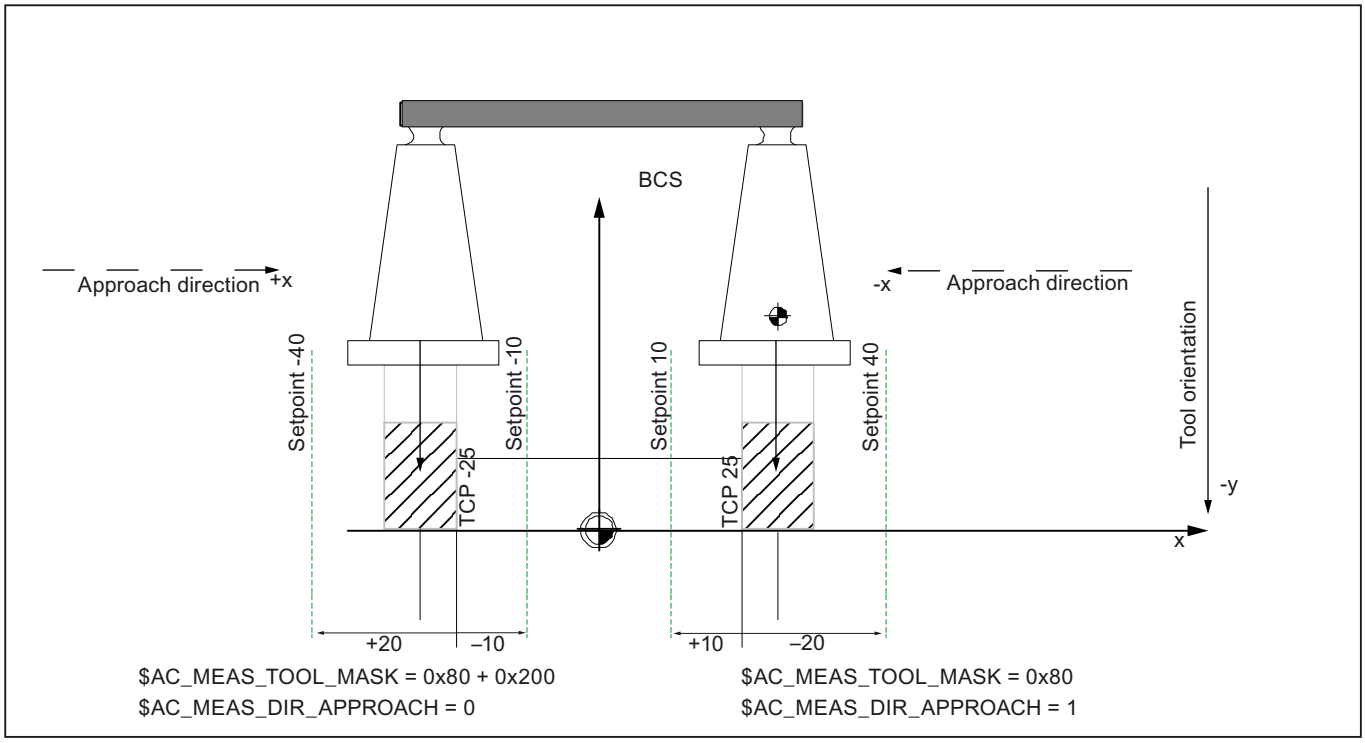
Two milling tools each with their own reference point with a tool orientation in -y direction



In the case of the tool position of two milling tools each with their own reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

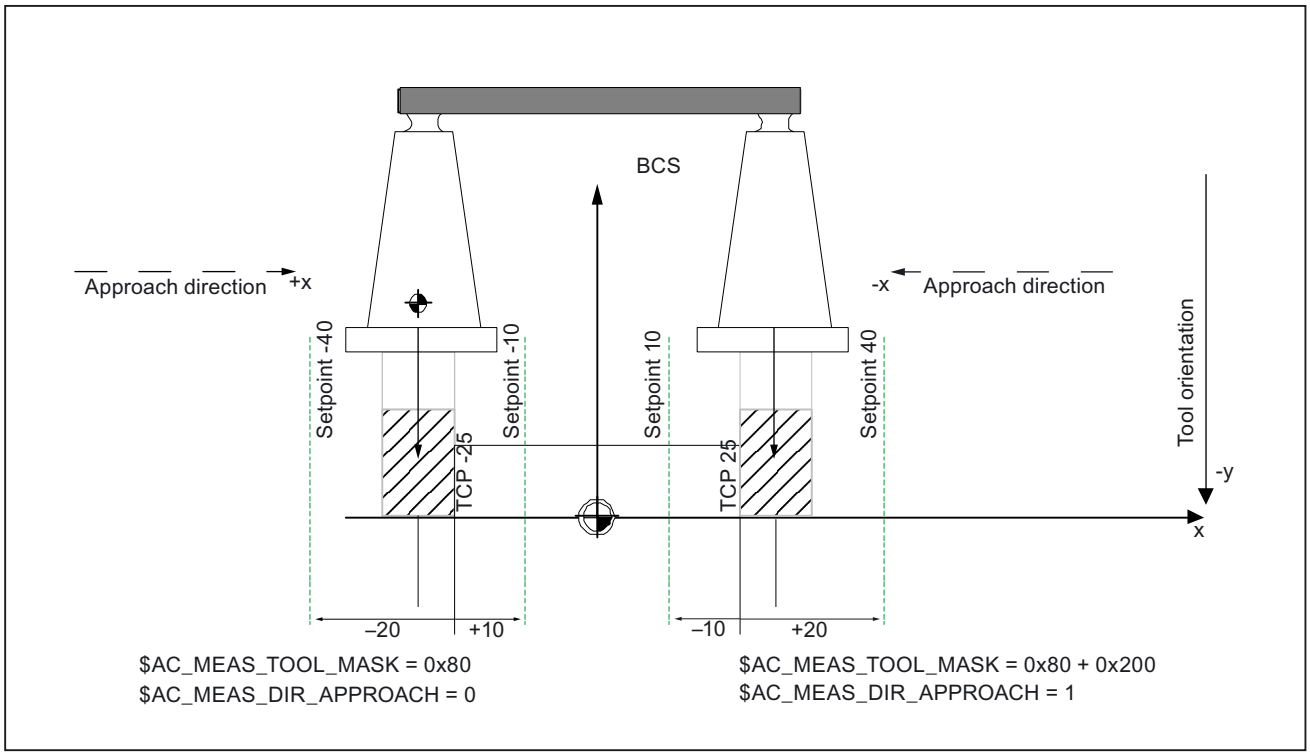
Approach direction +x, tool orientation -y	Approach direction -x, tool orientation -y
$AC_MEAS_TOOL_MASK = 0x80	$AC_MEAS_TOOL_MASK = 0x80
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1

Two milling tools with one reference point with a tool orientation in -y



In the case of the tool position of two milling tools with one reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

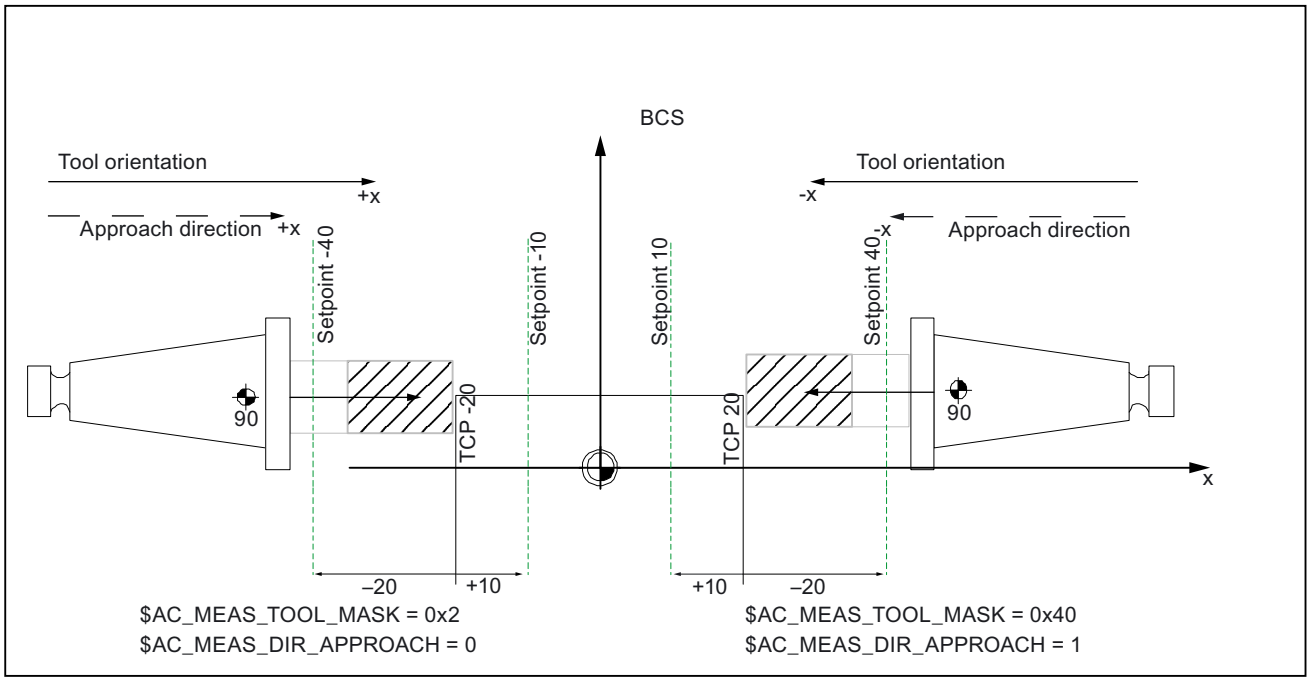
Approach direction +x, tool orientation -y	Approach direction -x, tool orientation -y
$AC_MEAS_TOOL_MASK = 0x80 + 0x200	$AC_MEAS_TOOL_MASK = 0x80
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1



In the case of the tool position of two milling tools with one reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

Approach direction +x, tool orientation -y	Approach direction -x, tool orientation -y
$AC_MEAS_TOOL_MASK = 0x80	$AC_MEAS_TOOL_MASK = 0x80 + 0x200
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1

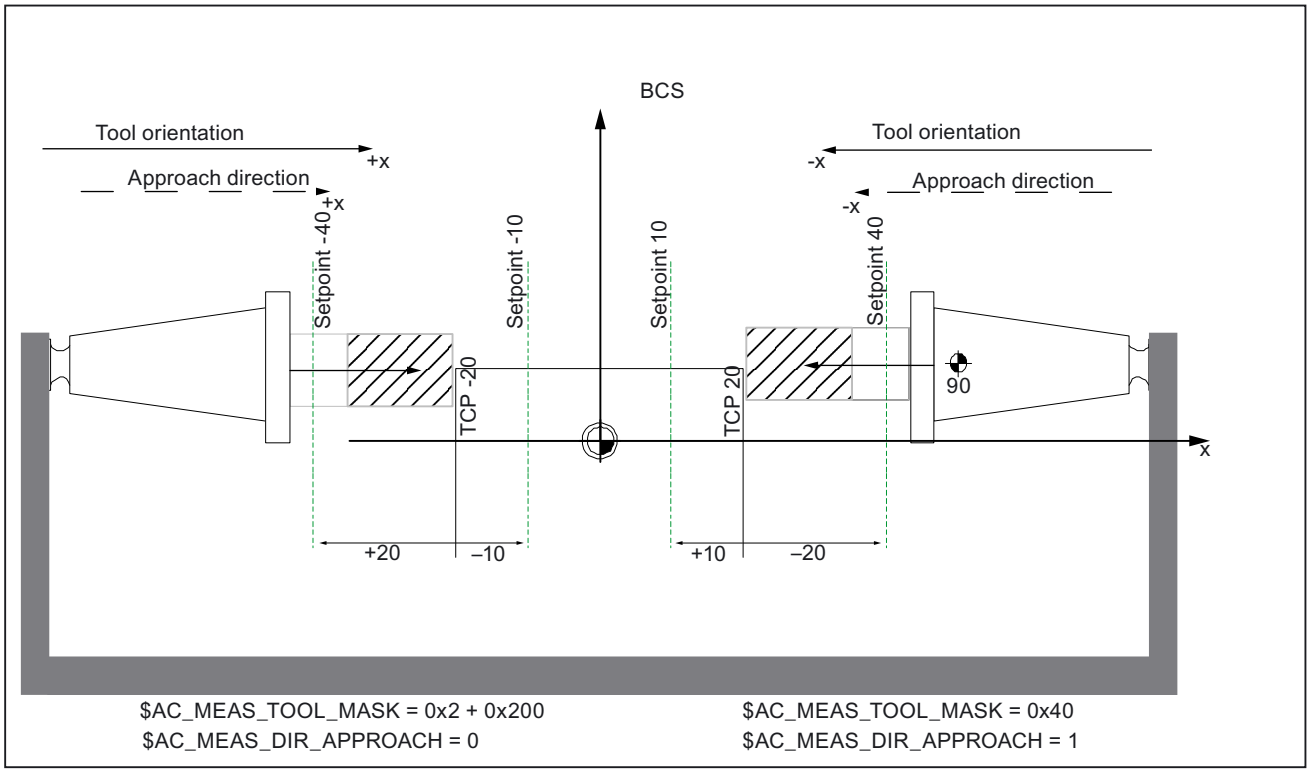
Two milling tools each with their own reference point with a tool orientation in the approach direction



In the case of the tool position of two milling tools each with their own reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

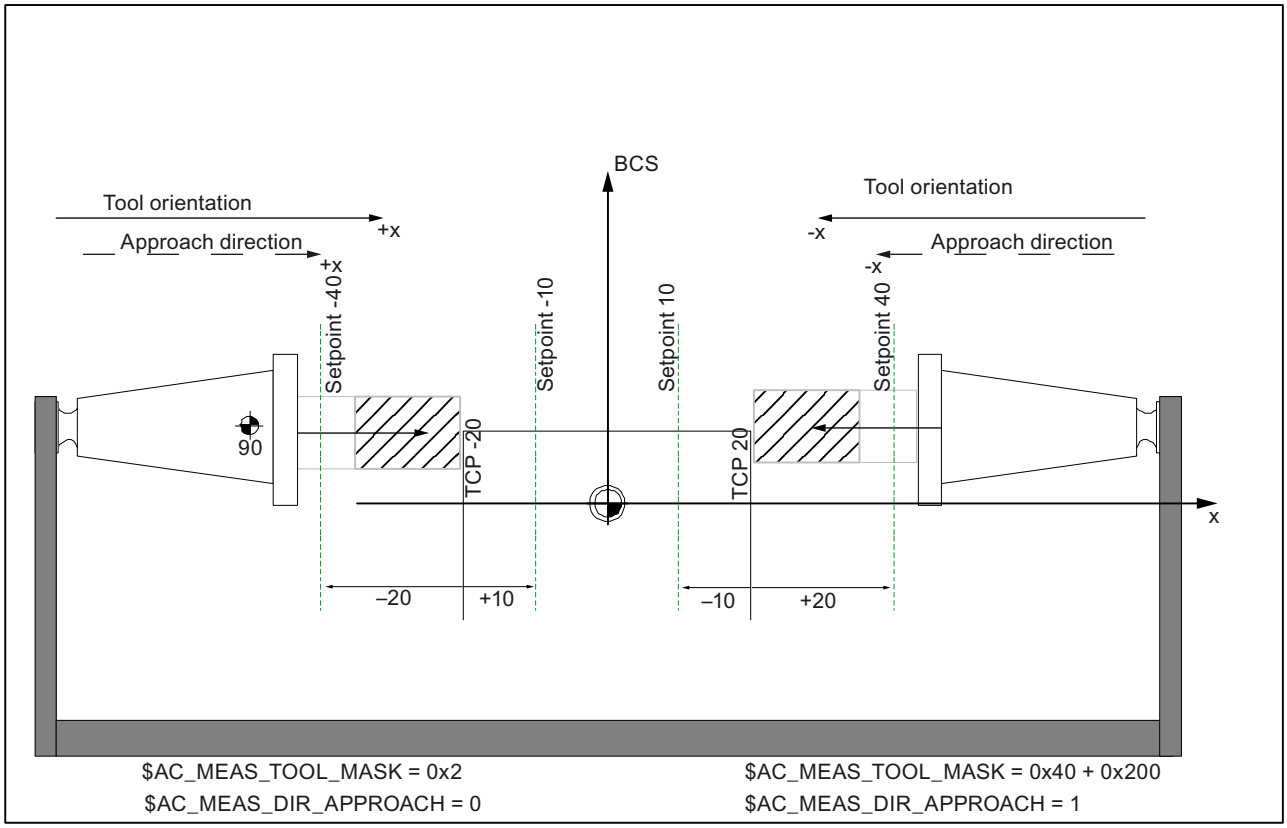
Approach direction and tool orientation +x	Approach direction and tool orientation -x
$AC_MEAS_TOOL_MASK = 0x2	$AC_MEAS_TOOL_MASK = 0x40
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1

Two milling tools with one reference point with a tool position opposite to the orientation



In the case of the tool position of two milling tools with one reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

Approach direction and tool orientation +x	Approach direction and tool orientation -x
$AC_MEAS_TOOL_MASK = 0x2 +200	$AC_MEAS_TOOL_MASK = 0x40
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1



In the case of the tool position of two milling tools with one reference point, not only are the input variables of measurement type 23 evaluated but also the values of the following input variables:

Approach direction and tool orientation +x	Approach direction and tool orientation -x
$AC_MEAS_TOOL_MASK = 0x2	$AC_MEAS_TOOL_MASK = 0x40 +200
$AC_MEAS_DIR_APPROACH = 0	$AC_MEAS_DIR_APPROACH = 1

Randomly oriented tools

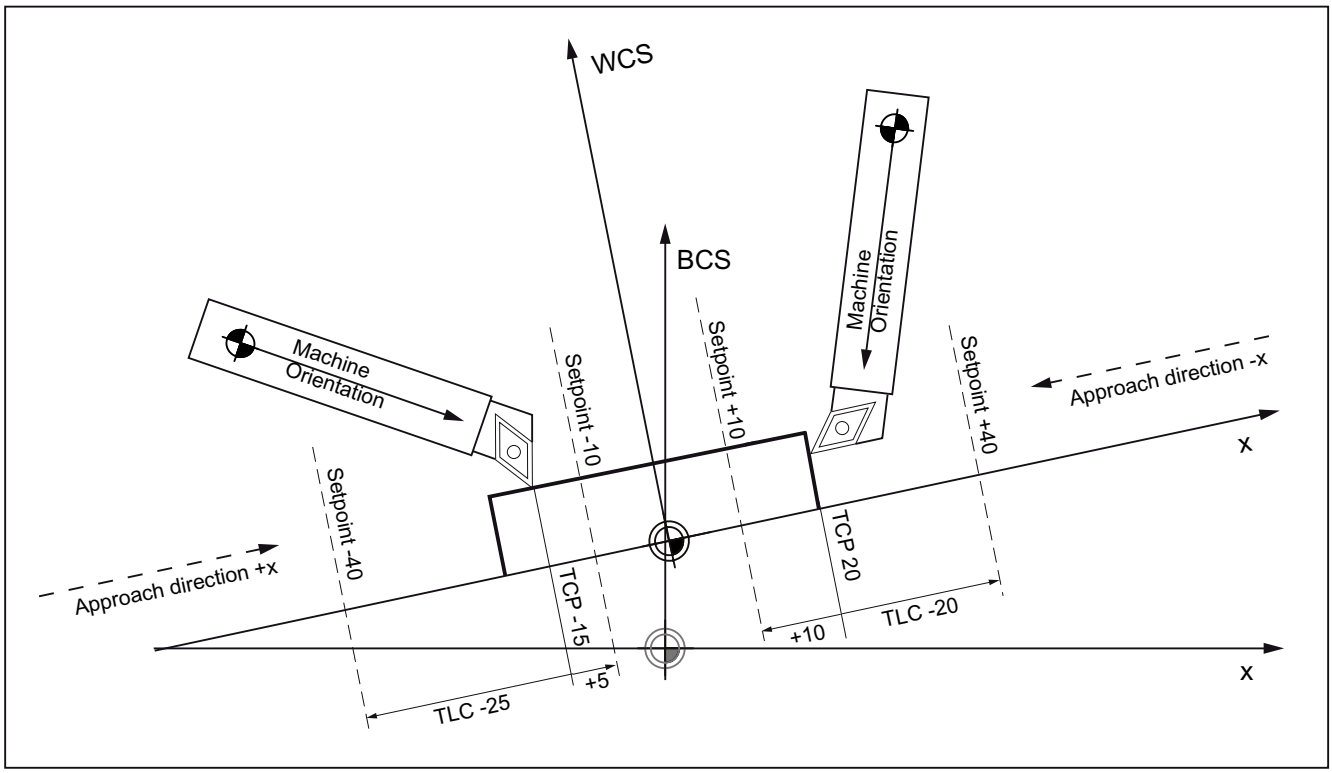


Figure 8-25 Two turning tools each with their own reference point

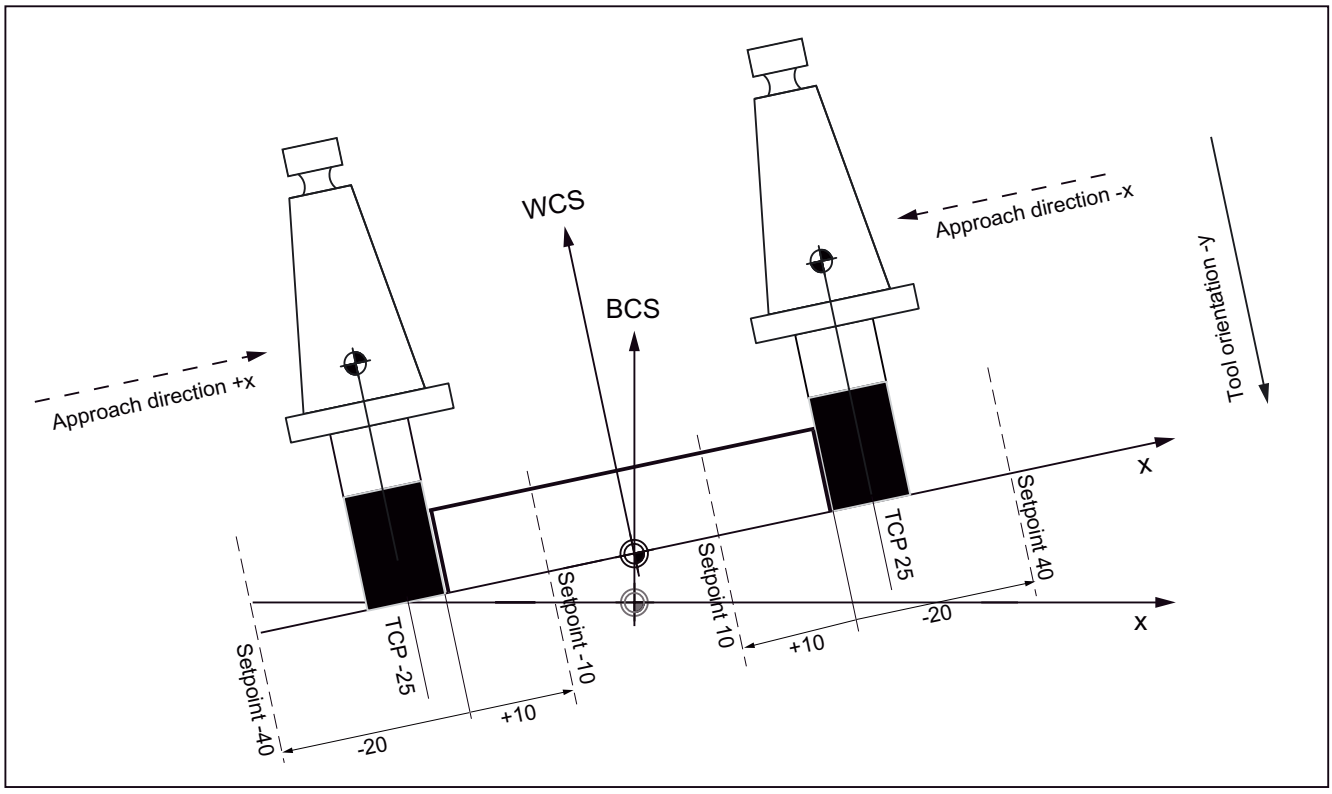


Figure 8-26 Two milling tools each with its own reference point

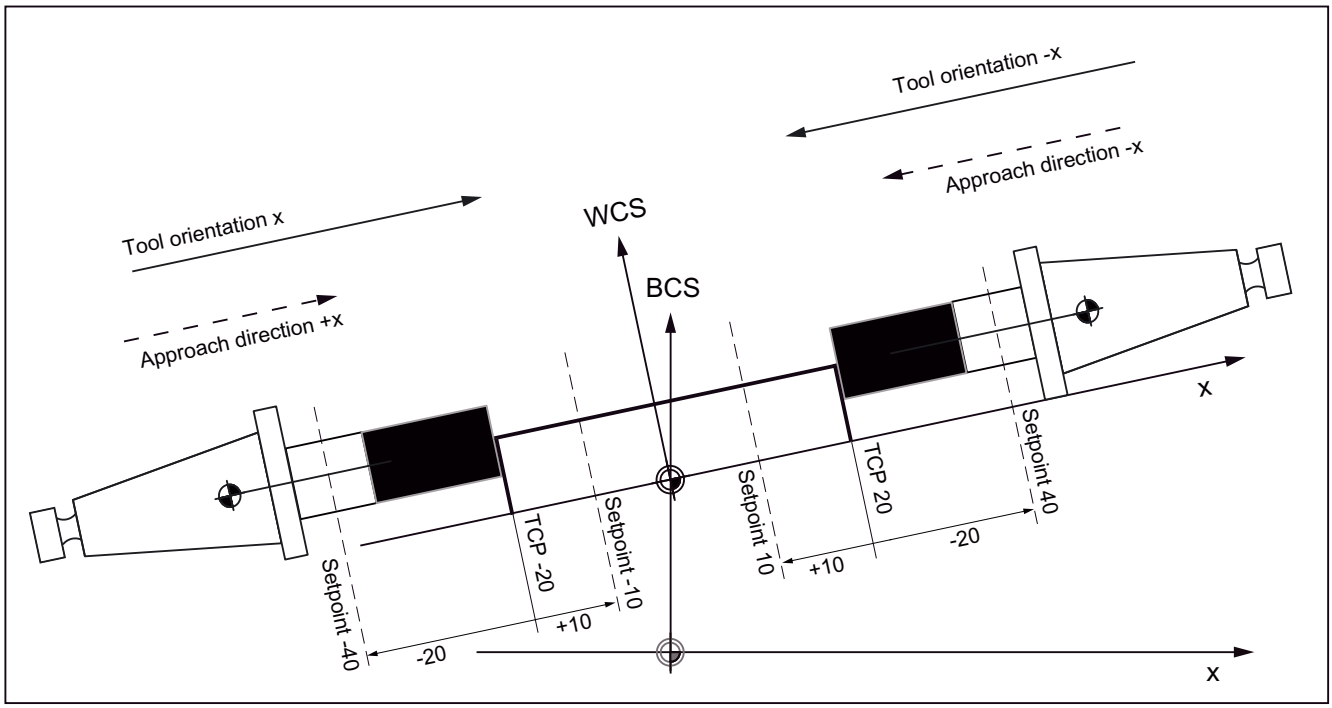


Figure 8-27 Two milling tools rotated at 90 degrees each with their own reference point

8.5 Measurement accuracy and functional testing

8.5.1 Measurement accuracy

Accuracy

The propagation time of the measuring signal is determined by the hardware used.
The delay times when using SIMODRIVE 611D are in the 3.625 µ ... 9.625 µ range plus the reaction time of the probe.

The measurement uncertainty is therefore:

Measurement uncertainty = propogation time of the measurement signal x traversing velocity

The allowed traversing velocities depend on the number of programmed measurement edges and the relation between the IPO/position control clock cycle.

Correct results only comply to traversing velocities for which not more than 1 equal and not more than 4 different trigger signals per position control clock cycle arrive.

8.5.2 Probe functional testing

Example of function test

```
%_N_PRUEF_MESSTASTER_MPF
;$PATH=/_N_MPF_DIR
;Testing program probe connection
N05 DEF INT MTSIGNAL                          ; Flag for trigger status
N10 DEF INT ME_NR=1                           ; measurement input number
N20 DEF REAL MESSWERT_IN_X
N30 G17 T1 D1                                 ; tool compensation for
                                              ; preselect probe
N40 _ANF: G0 G90 X0 F150                      ; starting position and
                                              ; measuring velocity
N50 MEAS=ME_NR G1 X100                        ; measurement at measurement input =1
                                              ; in the X axis
N60 STOPRE
N70 MTSIGNAL=$AC_MEA[1]                       ; read software switching signal
                                              ; at 1st measurement input
N80 IF MTSIGNAL == 0 GOTOF _FEHL1             ; Evaluation of signal
N90 MESSWERT_IN_X=$AA_MW[X]                   ; Read in measured value of
                                              ; workpiece coordinates
N95 M0
N100 M02
N110 _FEHL1: MSG ("Probe not switching!")
N120 M0
N130 M02
```

8.6 Examples - only 840D sl

8.6.1 Measuring mode 1

Measurement with one encoder

- One-time measurement
- One probe
- Trigger signals are the rising and falling edges
- Actual value from the current encoder

```
N2     MEASA[X] = (1, 1, -1) G01 X100 F100
N3     STOPRE
N4     IF $AC_MEA[1]==FALSE gotof ENDE
N5     R10=$AA_MM1[X]
N6     R11=$AA_MM2[X]
N7     END:
```

Measurement with two encoders

- One-time measurement
- One probe
- Trigger signals are the rising and falling edges
- Current values with two encoders

```
N2     MEASA[X] = (31, 1, -1) G01 X100 F100
N3     STOPRE
N4     IF $AC_MEA[1]==FALSE gotof ENDE
N5     R10=$AA_MM1[X]
N6     R11=$AA_MM2[X]
N7     R12=$AA_MM3[X]
N8     R13=$AA_MM4[X]
N9     END:
```

8.6.2 Measuring mode 2

- Two probes
- Trigger signals are the rising and falling edges
- Actual value from the current encoder

```
N2     MEASA[X] = (2, 1, -1, 2, -2) G01 X100 F100
N3     STOPRE
N4     IF $AC_MEA[1]==FALSE gotof MESSTASTER2
N5     R10=$AA_MM1[X]
N6     R11=$AA_MM2[X]
N7     PROBE2
N8     IF $AC_MEA[2]==FALSE gotof ENDE
N9     R12=$AA_MM3[X]
N10    R13=$AA_MM4[X]
N11    END:
```

8.6.3 Continuous measurement

8.6.3.1 Continous measurement on completion of programmed traversing motion

- The measurement is done in measuring mode 1:
- Measurement with 100 values
- One probe
- Trigger signal is the falling edge
- Actual value from the current encoder

```
N1     DEF REAL MESSWERT[100]
N2     DEF INT INDEX=0
N3     MEAC[x]=(1, 1, -1) G01 X1000 F100
N4     MEAC[X]=(0)                          ; Abort
N5     R1=$AC_FIFO1[4]                      ; Number of measured values
N6     FOR INDEX=0 TO R1
N7     MESSWERT[INDEX]=$AC_FIFO1[0]         ; Read out measured values
N8     ENDFOR:
```

8.6.3.2 Continuous measurements with deletion of distance-to-go

- Delete distance-to-go after last measurement
- The measurement is done in measuring mode 1:
- Measurement with 100 values
- One probe
- Trigger signal is the falling edge
- Actual value from the current encoder

```
N1    DEF INT ANZAHL=100
N2    DEF REAL MESSWERT[ANZAHL]
N3    DEF INT INDEX=0
N4    WHEN $AC_FIFO1[4]==ANZAHL DO DELDTG (X) MEAC[X] =(0)
N5    MEAC[X]=(1, 1, -1) G01 X1000 F100                    ; Start measurement
N6    R1=$AC_FIFO1[4]                                      ; Number of measured
                                                             values
N7    FOR INDEX=0 TO R1
N8    MESSWERT[INDEX]=$AC_FIFO1[0]                         ; Read out measured
                                                             values
N9    ENDFOR:
```

8.6.3.3 Continuous measurements modally over several blocks

- The measurement is done in measuring mode 1:
- Measurement with 100 values
- One probe
- Trigger signal is the falling edge
- Actual value from the current encoder

```
N1    DEF INT ANZAHL=100
N2    DEF REAL MESSWERT[ANZAHL]
N3    DEF INT INDEX=0
N4    ID=1 MEAC[X]=(1, 1, -1)                              ; Start
                                                             measurement
N5    ID=2 WHEN $AC_FIFO1[4]==ANZAHL DO MEAC[X]=(0) CANCEL(2)
N6    G01 X1000 Y100
N7    X100 Y100
N8    R1=$AC_FIFO1[4]                                      ; Number of
                                                             measured values
N9    FOR INDEX=0 TO R1
N10   MESSWERT[INDEX]=$AC_FIFO1[0]                         ; Read out
                                                             measured values
N11   ENDFOR:
```

8.6.4 Functional test and repeat accuracy

Function test

```
%_N_PRUEF_MESSTASTER_MPF
;$PATH=/_N_MPF_DIR
;Testing program probe connection
N05 DEF INT MTSIGNAL                          ; Flag for trigger status
N10 DEF INT ME_NR=1                           ; measurement input number
N20 DEF REAL MESSWERT_IN_X
N30 G17 T1 D1                                 ; tool compensation for
                                              ; preselect probe
N40 _ANF: G0 G90 X0 F150                      ; starting position and
                                              ; measuring velocity
N50 MEAS=ME_NR G1 X100                        ; measurement at measurement input =1
                                              ; in the X axis

N60 STOPRE
N70 MTSIGNAL=$AC_MEA[1]                       ; read software switching signal
                                              ; at 1st measurement input
N80 IF MTSIGNAL == 0 GOTOF _FEHL1             ; evaluation of signal
N90 MESSWERT_IN_X=$AA_MW[X]                   ; read in measured value of
                                              ; workpiece coordinates
N95 M0
N100 M02
N110 _FEHL1: MSG ("Probe not switching!")
N120 M0
N130 M02
```

Repeat accuracy

This program allows the measuring scatter (repeat accuracy) of the entire measuring system (machine-probe-signal transmission to NC) to be calculated.

In the example, ten measurements are taken in the X axis and the measured value recorded in the workpiece coordinates.

It is therefore possible to determine the random dimensional deviations which are not subject to any trend.

```
%_N_PRUEF_GENAU_MPF;
$PATH=/_N_MPF_DIR
N05 DEF INT SIGNAL, II                    ; Variable definition
N10 DEF REAL MESSWERT_IN_X[10]
```

```
N15 G17 T1 D1                              ; Initial conditions,
                                           : Tool compensation
                                           ; preselect for probe
N20 _ANF: G0 X0 F150 ←                     ; Prepositioning in the measured axis
N25 MEAS=+1 G1 X100 ←                      ; at 1st measurement input when
                                           ; switching signal not deflected,
                                           ; deflected in the X axis
N30 STOPRE ←                               ; Stop decoding for this after
                                           ; subsequent evaluation of
                                           ; results
N35 SIGNAL= $AC_MEA[1]                     ; read software switching signal at
                                           ; 1st measurement input
N37 IF SIGNAL == 0 GOTOF_FEHL1             ; Check switching signal
N40 MESSWERT_IN_X[II]=$AA_MW[X]            ; Read measured value in workpiece
                                             coordinates
N50 II=II+1
N60 IF II<10 GOTOB_ANF                     ; Repeat 10 times
N65 M0
N70 M02
N80 _FEHL1: MSG ("Probe not switching")
N90 M0
N95 M02
```

After the parameter display (user-defined variables) have been selected, the measurement results can be read in field MEASVALUE_IN_X[10] provided that the program is still being processed.

8.7 Data lists

8.7.1 Machine data

8.7.1.1 General machine data

Number	Identifier: $MN_	Description
13200	MEAS_PROBE_LOW_ACTIVE	Switching characteristics of probe
13201	MEAS_PROBE_SOURCE	Measurement pulse simulation via digital output
13210	MEAS_TYPE	Type of measurement for PROFIBUS DP drives

8.7.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20360	TOOL_PARAMETER_DEF_MASK	Definition of tool parameters
28264	MM_LEN_AC_FIFO	Length of $AC_FIFO ... FIFO variables

8.7.2 System variables

Table of all the input values

Type	System variable name	Description
INT	$AC_MEAS_SEMA	Interface assignment
INT	$AC_MEAS_VALID	Validity bits for input values
REAL	$AA_MEAS_POINT1[ax]	1st measuring point for all channel axes
REAL	$AA_MEAS_POINT2[ax]	2nd measuring point for all channel axes
REAL	$AA_MEAS_POINT3[ax]	3rd measuring point for all channel axes
REAL	$AA_MEAS_POINT4[ax]	4th measuring point for all channel axes
REAL	$AA_MEAS_SETPOINT[ax]	Setpoint position for all channel axes
REAL	$AA_MEAS_SETANGLE[ax]	Setpoint angle for all channel axes
INT	$AC_MEAS_P1_COORD	Coord. system for the 1st measuring point
INT	$AC_MEAS_P2_COORD	Coord. system for the 2nd measuring point
INT	$AC_MEAS_P3_COORD	Coord. system for the 3rd measuring point
INT	$AC_MEAS_P4_COORD	Coord. system for the 4th measuring point
INT	$AC_MEAS_SET_COORD	Coordinate system of setpoint
INT	$AC_MEAS_LATCH[0..3]	Pick up measuring points in the WCS
INT	$AA_MEAS_P1_VALID[ax]	Pick up 1st measuring point in the WCS

Type	System variable name	Description
INT	$AA_MEAS_P2_VALID[ax]	Pick up 2nd measuring point in the WCS
INT	$AA_MEAS_P3_VALID[ax]	Pick up 3rd measuring point in the WCS
INT	$AA_MEAS_P4_VALID[ax]	Pick up 4th measuring point in the WCS
INT	$AA_MEAS_SP_VALID[ax]	Set setpoint position of axis as valid
REAL	$AC_MEAS_WP_SETANGLE	Setpoint workpiece position angle
REAL	$AC_MEAS_CORNER_SETANGLE	Setpoint cutting angle of corner
INT	$AC_MEAS_DIR_APPROACH	Approach direction
INT	$AC_MEAS_ACT_PLANE	Working plane of tool
INT	$AC_MEAS_SCALEUNIT	Unit of measurement INCH / METRIC
INT	$AC_MEAS_FINE_TRANS	Corrections in fine displacement
INT	$AC_MEAS_FRAME_SELECT	Frame selection during tool measurement
INT	$AC_MEAS_CHSFR	Frame chain setting: System frames
INT	$AC_MEAS_NCBFR	Frame chain setting: Global basic frames
INT	$AC_MEAS_CHBFR	Frame chain setting: Channel basic frames
INT	$AC_MEAS_UIFR	Frame chain setting: Settable frames
INT	$AC_MEAS_PFRAME	Frame chain setting: Programmed frame
INT	$AC_MEAS_T_NUMBER	Tool selection
INT	$AC_MEAS_D_NUMBER	Cutting edge selection
INT	$AC_MEAS_TOOL_MASK	Tool settings
INT	$AC_MEAS_TYPE	Measuring type
REAL	$AC_MEAS_INPUT[10]	Measurement input parameters

Table of all the output values

Type	System variable name	Description
FRAME	$AC_MEAS_FRAME	Result frame
REAL	$AC_MEAS_WP_ANGLE	Calculated workpiece position angle
REAL	$AC_MEAS_CORNER_ANGLE	Calculated angle of intersection
REAL	$AC_MEAS_DIAMETER	Calculated diameter
REAL	$AC_MEAS_TOOL_LENGTH	Calculated tool length
REAL	$AC_MEAS_RESULTS[n]	Measurement results (depending on measurement type)

N3: Software cams, position switching cycles - only 840D sl

9

9.1 Brief description

Function

The "Software cams" function generates position-dependent switching signals for axes that supply an actual position value (machine axes) and for simulated axes.
These cam signals can be output to the PLC and also to the NCK I/Os.

The cam positions at which signal outputs are set can be defined and altered via setting data. The setting data can be read and written via HMI, PLC and part program.

Activation

The "Software cams" function can be activated and used in all operating modes.
The function remains active in the event of `RESET` or EMERGENCY STOP.

Field of application

Examples of cam signal applications are as follows:

- To activate protection zones
- To initiate additional movements as a function of position
- As reversing signals for hydraulically controlled oscillation axes

Axis types

Software cams can be used on linear and modulo rotary axes that are defined as machine axes.

Cam range/cam pair

Cams are always assigned in pairs to axes. A pair consists of a plus and a minus cam. 32 cam pairs are available.

The plus and minus cams each simulate a mechanical cam which is actuated at a defined point (cam position) in a specific approach direction when the axis reaches the cam position.

Cam ranges are assigned to the plus and minus cams as follows:

Cam range plus:	All positions ≥ plus cam
Cam range minus:	All positions ≤ minus cam

9.2 Cam signals and cam positions

9.2.1 Generation of cam signals for separate output

General information

Both cam signals can be output to the PLC and to the NCK I/Os. Separate output of the plus and minus cam signals makes it easy to detect whether the axis is within or outside the plus or minus cam range.

Linear axes

The switching edges of the cam signals are generated as a function of the axis traversing direction:

- The minus cam signal switches from 1 to 0 when the axis traverses the minus cam in the positive axis direction.
- The plus cam signal switches from 0 to 1 when the axis traverses the plus cam in the positive direction.

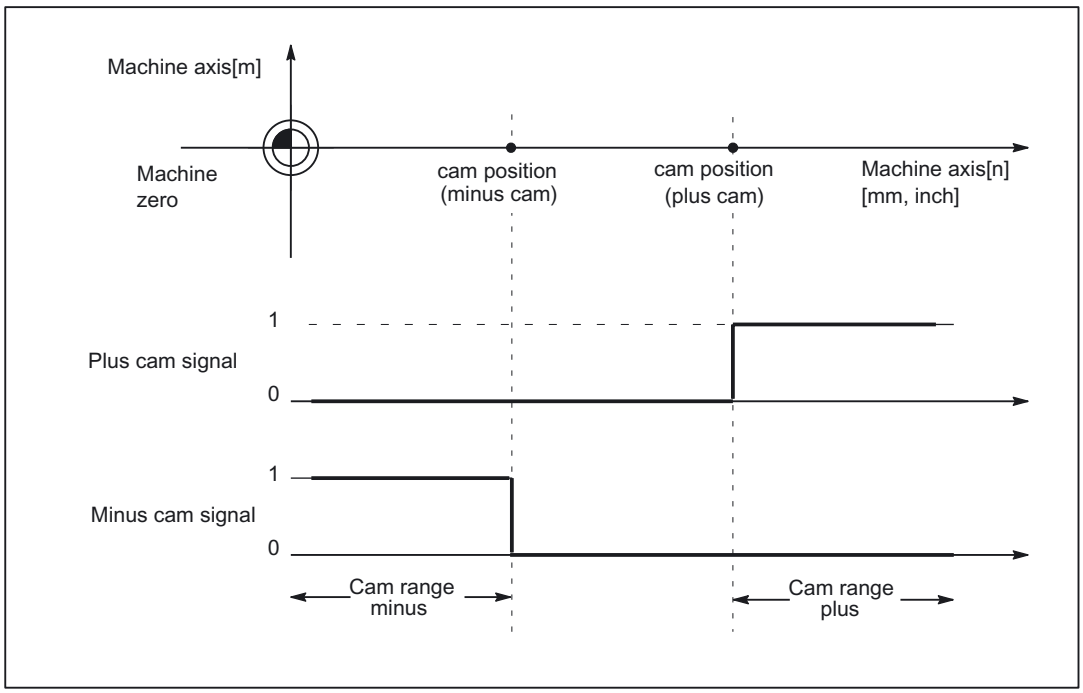


Figure 9-1 Software cams for linear axis (minus cam < plus cam)

Note

Position switching signals:

If the axis is positioned exactly on the cam (plus or minus), the defined output flickers. If the axis moves one increment further, the output becomes a definite zero or one.

Flickering of the actual position causes the signals to flicker in this manner. The actual position is evaluated.

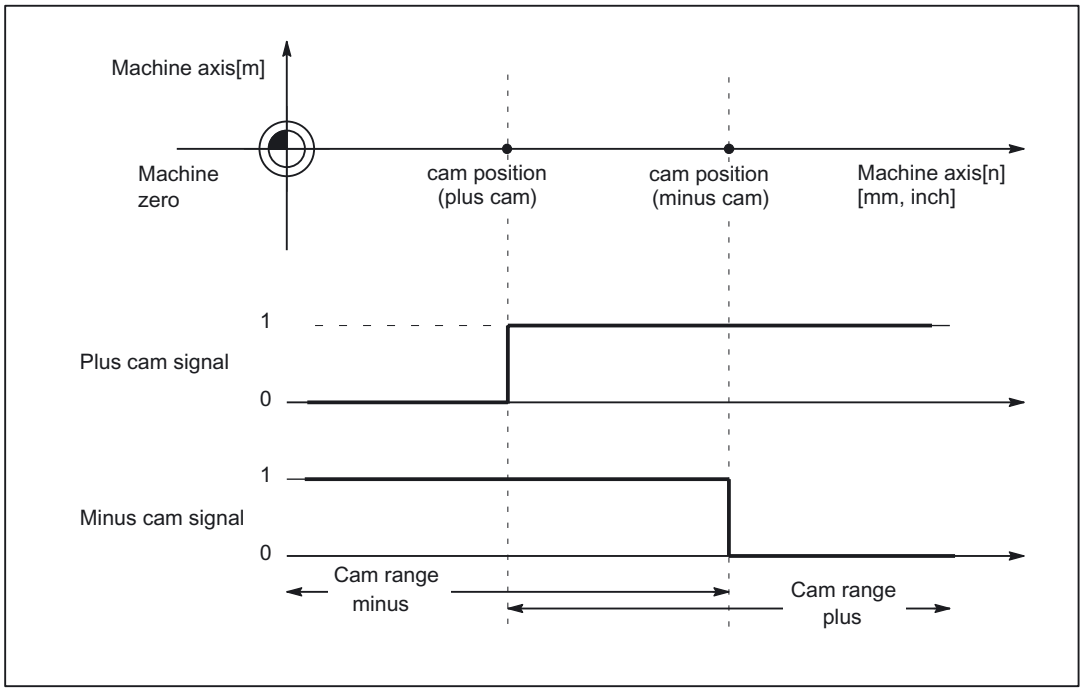


Figure 9-2 Software cams for linear axis (plus cam < minus cam)

Modulo rotary axes

The switching edges of the cam signals are generated as a function of the rotary axis traversing direction:

- The plus cam signal switches from 0 to 1 when the axis traverses the minus cam in a positive axis direction and from 1 back to 0 when it traverses the plus cam.
- The minus cam signal changes level in response to every positive edge of the plus cam signal.

Note

The described response of the plus cam applies on **condition** that:

plus cam - minus cam < 180 degrees

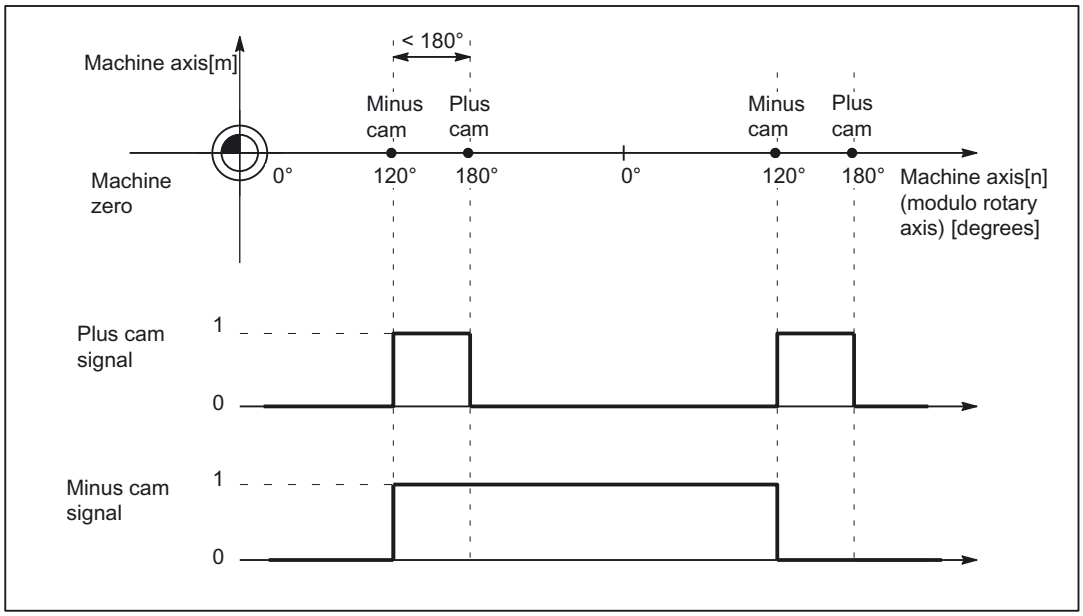


Figure 9-3 Software cams for modulo rotary axis (plus cam - minus cam < 180°)

The signal change of the minus cam makes it possible to detect traversal of the cam even if the cam range is set so small that the PLC cannot detect it reliably.

Both cam signals can be output to the PLC and to the NCK I/Os. Separate output of the plus and minus cam signals makes it easy to detect whether the axis is within or outside the plus or minus cam range.

If this condition (plus cam - minus cam < 180°) is not fulfilled or if the minus cam is set to a greater value than the plus cam, then the response of the plus cam signal is inverted. The response of the minus cam signal remains unchanged.

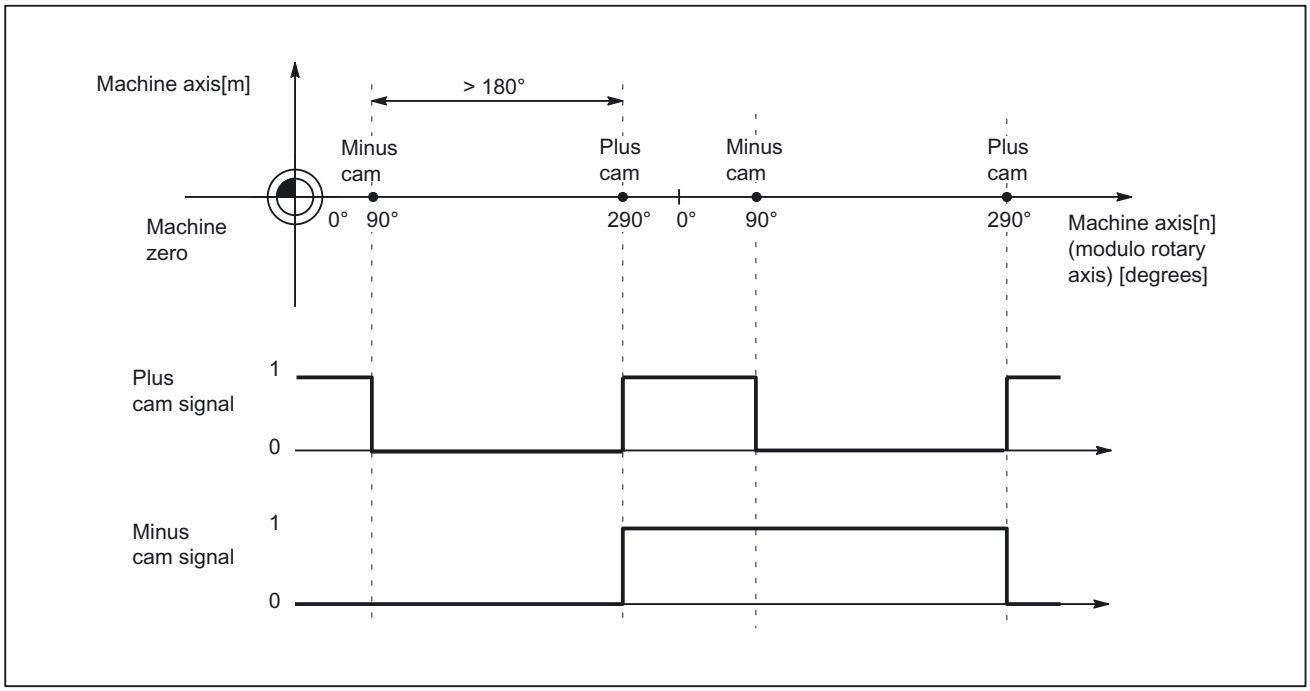


Figure 9-4 Software cams for modulo rotary axis (plus cam - minus cam > 180 degrees)

9.2.2 Generation of cam signals with gated output

General information

The plus and minus cam output signals are gated in the case of:

- timer-controlled cam signal output to the four onboard outputs on the NCU
- output to the NCK I/O devices if the 2nd byte in the following machine data was not specified (= "0"):
  MD10470 SW_CAM_ASSIGN_FASTOUT_2
  to
  MD10473 SW_CAM_ASSIGN_FASTOUT_4

Linear axes

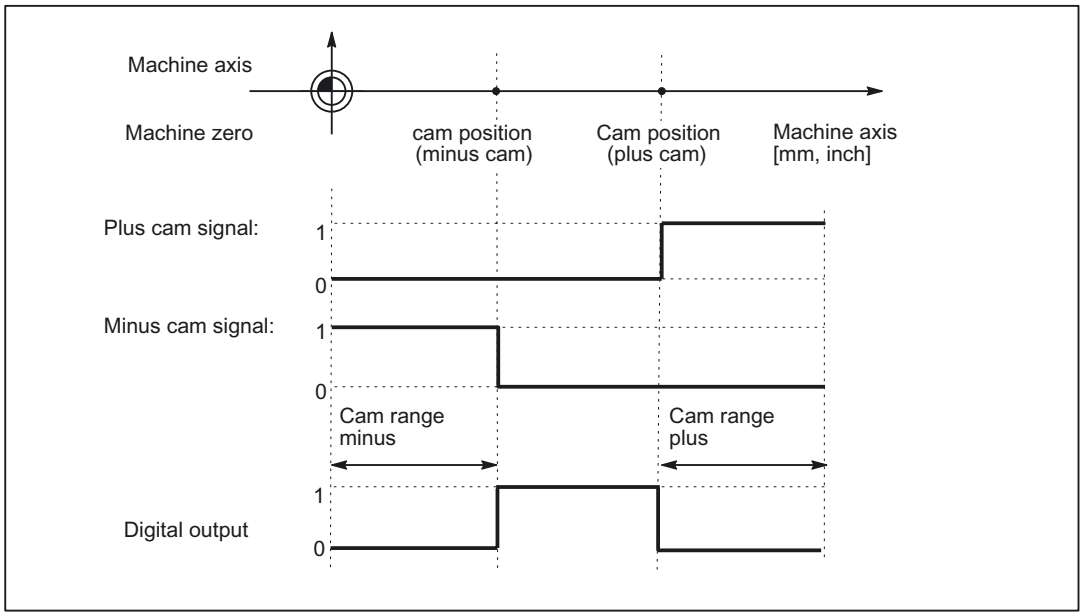


Figure 9-5 Position switching signals for linear axis (minus cam < plus cam)

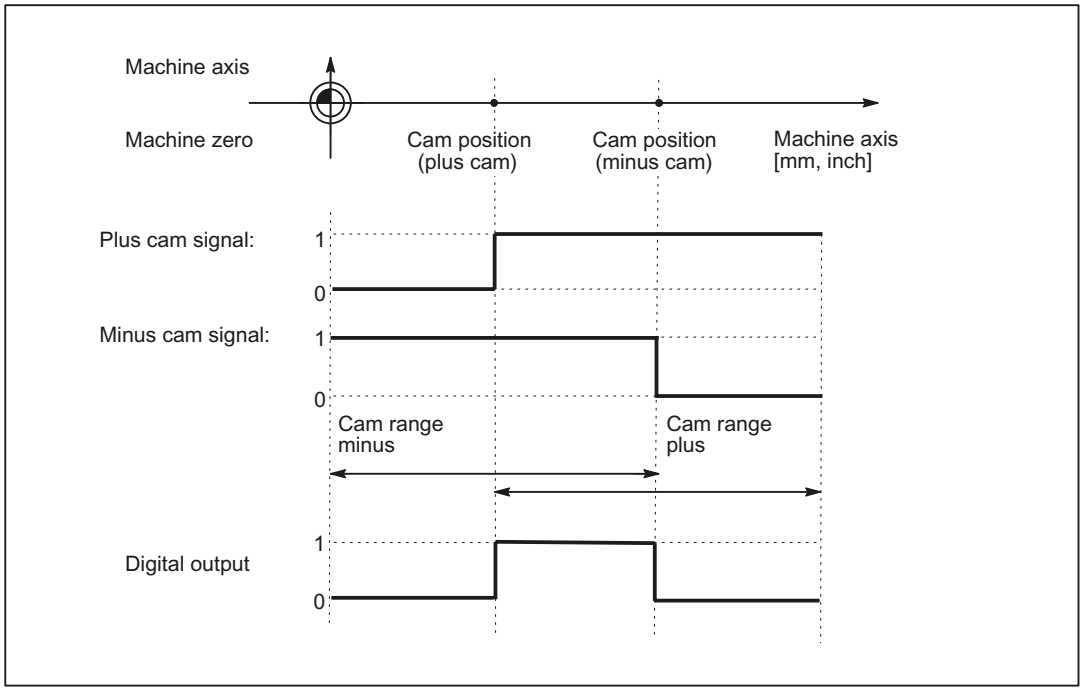


Figure 9-6 Position switching signals for linear axis (plus cam < minus cam)

Modulo rotary axis

The default signal response for modulo rotary axes is dependent on the cam width:

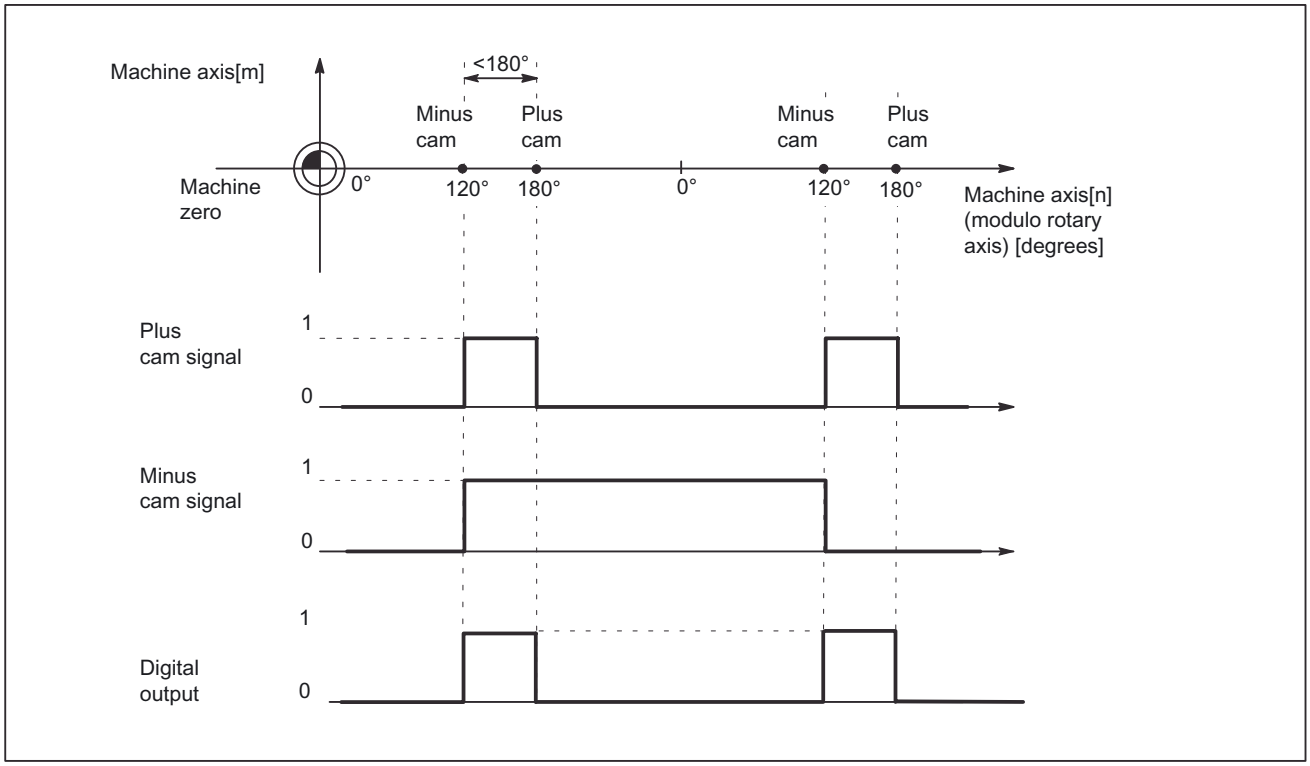


Figure 9-7 Software cams for modulo rotary axis (plus cam - minus cam < 180 degrees)

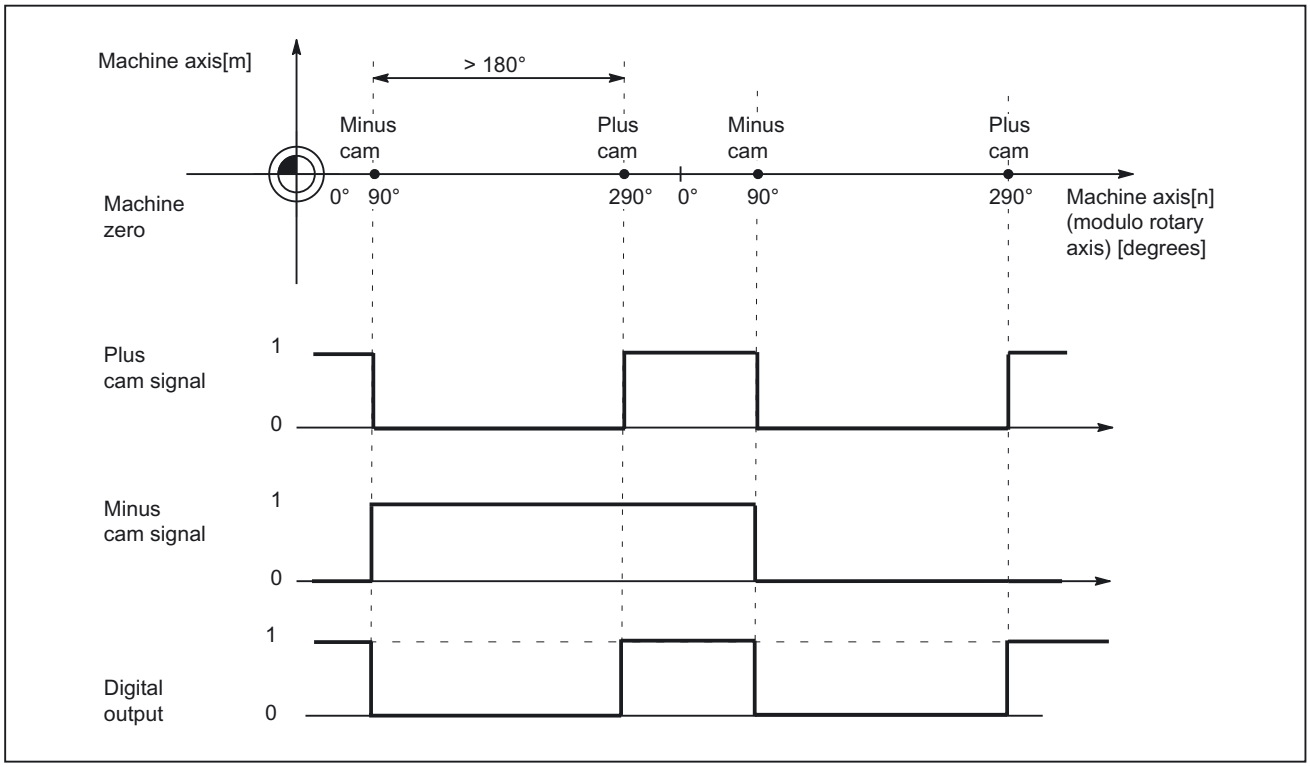


Figure 9-8 Software cams for modulo rotary axis (plus cam - minus cam > 180°)

Suppression of signal inversion

Machine data setting:
MD10485 SW_CAM_MODE Bit 1=1
can be used to select the suppression of signal inversion for:
plus cam - minus cam > 180°.

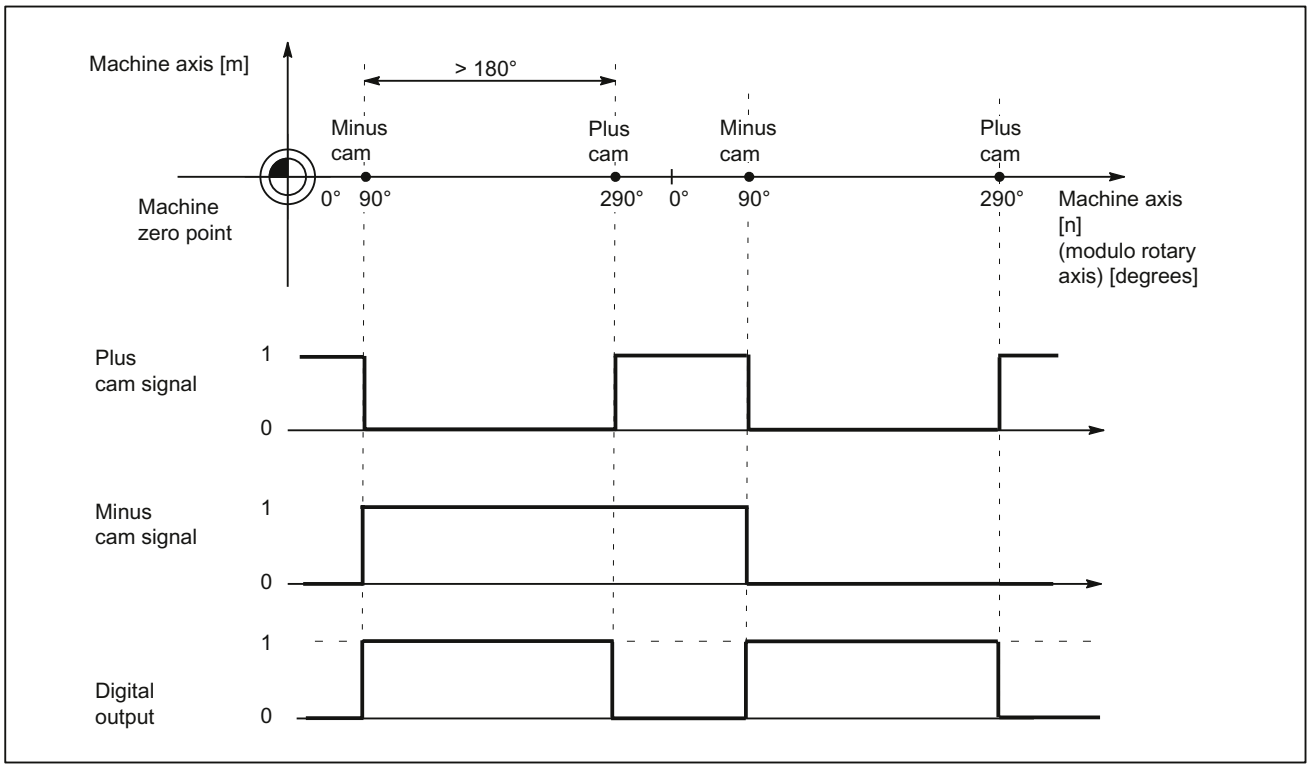


Figure 9-9 Software cams for modulo rotary axis (plus cam - minus cam > 180°) and suppression of signal inversion

9.2.3 Cam positions

Setting cam positions

The cam positions of the plus and minus cams are defined via the following general setting data:

SD41500 SW_CAM_MINUS_POS_TAB_1[n]	Position of minus cams 1 - 8
SD41501 SW_CAM_PLUS_POS_TAB_1[n]	Position of plus cams 1 – 8
SD41502 SW_CAM_MINUS_POS_TAB_2[n]	Position of minus cams 9 - 16
SD41503 SW_CAM_PLUS_POS_TAB_2[n]	Position of plus cams 9 - 16
SD41504 SW_CAM_MINUS_POS_TAB_3[n]	Position of minus cams 17 - 24
SD41505 SW_CAM_PLUS_POS_TAB_3[n]	Position of plus cams 17 - 24
SD41506 SW_CAM_MINUS_POS_TAB_4[n]	Position of minus cams 25 - 32
SD41507 SW_CAM_PLUS_POS_TAB_4[n]	Position of plus cams 25 - 32

Note

Owing to the grouping of cam pairs (eight in each group), it is possible to assign different access authorization levels (e.g. for machine-related and workpiece-related cam positions). The positions are entered in the machine coordinate system. No check is made with respect to the maximum traversing range.

Dimension system metric/inch

MD10260 CONVERT_SCALING_SYSTEM=1
With the above machine data setting, the cam positions no longer refer to the basic system, but to the system configured in machine data:
MD10270 POS_TAB_SCALING_SYSTEM

MD10270 POS_TAB_SCALING_SYSTEM=0:	Metric
MD10270 POS_TAB_SCALING_SYSTEM=1:	inch

MD10270 POS_TAB_SCALING_SYSTEM
The above machine data therefore defines the dimension system for position parameters from setting data:
SD41500 to SD41507

A switchover with G70/G71 or G700/G710 has no effect.

Sensing of cam positions

To set the cam signals, the actual position of the axes is compared to the cam position.

Writing/reading of cam positions

The setting data can be read and written via HMI, PLC and part program.

Accesses from the part program are not synchronous to machining.
Synchronization can only be achieved by means of a programmed block preprocessing stop (STOPRE command).

It is possible to read and write the cam positions with FB2 and FB3 in the PLC user program.

Axis/cam assignment

A cam pair is assigned to a machine axis using general machine data:
MD10450 SW_CAM_ASSIGN_TAB[n]
(assign software cams to machine axes).

Note

Changes to an axis assignment take effect after the next NCK power-up.

Cam pairs to which no axis is assigned are not active.

A cam pair can only be assigned to one machine axis at a time.

Several cam pairs can be defined for one machine axis.

9.2.4 Lead/delay times (dynamic cam)

Times

To compensate for any delays, it is possible to assign two lead or delay times with additive action to each minus and plus cam for the cam signal output.

The two lead or delay times are entered in a machine data and a setting data.

Note

The input of negative time values causes a delay in the output of cam signals.

Input in machine data

The **first** lead or delay time is entered in the following general machine data:

MD10460 SW_CAM_MINUS_LEAD_TIME[n]
(lead or delay time on minus cams)

MD10461 SW_CAM_PLUS_LEAD_TIME[n]
(lead or delay time on plus cams)

The following can be entered in these machine data, for example:

- Constant internal delay times between actual-value sensing and cam signal output (e.g. as determined by an oscilloscope)
- Constant external delay times

Input in setting data

The second lead or delay time is entered in the following general setting data:

SD41520 SW_CAM_MINUS_TIME_TAB_1[n]	Lead or delay time on minus cams 1 - 8
SD41521 SW_CAM_PLUS_TIME_TAB_1[n]	Lead or delay time on plus cams 1 - 8
SD41522 SW_CAM_MINUS_TIME_TAB_2[n]	Lead or delay time on minus cams 9 - 16
SD41523 SW_CAM_PLUS_TIME_TAB_2[n]	Lead or delay time on plus cams 9 - 16
SD41524 SW_CAM_MINUS_TIME_TAB_3[n]	Lead or delay time on minus cams 17 - 24
SD41525 SW_CAM_PLUS_TIME_TAB_3[n]	Lead or delay time on plus cams 17 - 24
SD41526 SW_CAM_MINUS_TIME_TAB_4[n]	Lead or delay time on minus cams 25 - 32
SD41527 SW_CAM_PLUS_TIME_TAB_4[n]	Lead or delay time on plus cams 25 - 32

Delay times which may change during machining must, for example, be entered in these setting data.

9.3 Output of cam signals

9.3.1 Activating

The status of the cam (cam signals) can be output to the PLC as well as to the NCK I/Os.

Activation of cam signal output

The output of cam signals for an axis is activated via
axis-specific NC/PLC interface signal:
DB31, ... DBX2.0 (cam activation).

Check-back signal to PLC

The successful activation of all cams of an axis is signaled back to the PLC via
axis-specific NC/PLC interface signal:
DB31, ... DBX62.0 (cam active).

Note

The activation can be associated with other conditions by the PLC user (e.g. axis referenced, RESET active).

9.3.2 Output of cam signals to PLC

Output to PLC

The status of the cam signals for all machine axes with activated software cams is output to the PLC.

The status is output in the IPO cycle and is transferred to the PLC asynchronously.

Minus cam signals

The status of the minus cam signals is entered in the general
NC/PLC interface signals:
DB10 DBX110.0 to 113.7 (minus cam signals 1 to 32).

Plus cam signals

The status of the plus cam signals is entered in the general NC/PLC interface signals:
DB10 DBX114.0 to 117.7 (plus cam signals 1 to 32).

If no measuring system is selected or NC/PLC interface signal:
DB31, ... DBX2.0 (cam activation)
is set to 0,
then the following NC/PLC interface signals are also set to 0:

DB10 DBX110.0-113.7 (minus cam signals 1-32)

DB10 DBX114.0-117.7 (plus cam signals 1-32)

DB 31, ... DBX62.0 (Cam active)

9.3.3 Output of cam signals to NCK I/Os in position control cycle

Signal output in position control cycle

For the cams assigned to a hardware byte in machine data:
MD10470 SW_CAM_ASSIGN_FASTOUT_1
to
MD10473 SW_CAM_ASSIGN_FASTOUT_4,
the signal output takes place in the position control cycle.

The 4 onboard outputs on the NCU and a total of 32 optional external NCK outputs are available as the digital outputs of the NCK I/Os.

References:
/FB2/ Function Manual, Extended Functions; Digital and Analog NCK I/Os (A4)

Hardware assignment

The assignment to the hardware bytes used is made for each eight cam pairs in the following general machine data:

MD10470 SW_CAM_ASSIGN_FASTOUT_1	Hardware assignment for output of cams 1 - 8 to NCK I/Os
MD10471 SW_CAM_ASSIGN_FASTOUT_2	Hardware assignment for output of cams 9 - 16 to NCK I/Os
MD10472 SW_CAM_ASSIGN_FASTOUT_3	Hardware assignment for output of cams 17 - 24 to NCK I/Os
MD10473 SW_CAM_ASSIGN_FASTOUT_4	Hardware assignment for output of cams 25 - 32 to NCK I/Os

Note

It is possible to define one HW byte for the output of eight minus cam signals and one HW byte for the output of eight plus cam signals in each machine data.

In addition, the output of the cam signals can be inverted with the two machine data.

If the 2nd byte is not specified (= "0"), then the 8 cams are output as a logic operation of the minus and plus cam signals via the 1st HW byte using the 1st inversion screen form.

Status query in the part program

The status of the HW outputs can be read in the part program with main run variable **$A_OUT[n]** (n = no. of output bit).

Switching accuracy

Signals are output to the NCK I/Os or onboard outputs in the position control cycle. Owing to the time scale of the position control cycle, the switching accuracy of the cam signals is limited as a function of velocity.

The following applies:

Delta pos = V_{act} * position control cycle

Delta pos:	Switching accuracy (depending on position control cycle)
V_{act}:	Current axis velocity

Example:

V_{act} = 20 m/min, PC cycle = 4 ms	Delta pos = 1.33 mm
V_{act} = 2000 rpm, PC cycle = 2 ms	Delta pos = 24 degrees

9.3.4 Timer-controlled cam signal output

Timer-controlled output

A significantly higher degree of accuracy can be achieved by outputting the cam signals independently of the position control cycle using a timer interrupt.

The following general machine data can be set to select timer-controlled output to the 4 NCU onboard outputs for 4 cam pairs:
MD10480 SW_CAM_TIMER_FASTOUT_MASK
(screen form for the output of cam signals via timer interrupts on NCU)

In this case, the minus and plus signals of a cam pair are gated for output as one signal.

Signal generation

The following machine data must be used beforehand to specify how the signals are to be gated:
MD10485 SW_CAM_MODE Bit 1

Bit	Signal generation
Not set	Inversion of signal response of plus cam when: plus cam - minus cam ≥ 180 degrees
set	**No** inversion of signal response of plus cam: plus cam - minus cam ≥ 180 degrees

Note

This function runs independently of the assignment in machine data:
MD10470 SW_CAM_ASSIGN_FASTOUT_1 or
MD10471 SW_CAM_ASSIGN_FASTOUT_2 or
MD10472 SW_CAM_ASSIGN_FASTOUT_3 or
MD10473 SW_CAM_ASSIGN_FASTOUT_4.

The onboard byte may not be used more than once at any one time.

Restrictions

The following applies to the mutual position of the cam positions:

Only **one** timer-controlled output takes place per interpolation cycle.

If signal changes for more than one cam pair are active in the same interpolation cycle, the output is prioritized:

The cam pair with the lowest number (1...32) determines the output time for **all** active signals, i.e. the signal change of the other cam pairs takes place at the same time.

PLC interface

The NCK image of the onboard outputs and the status of the plus and minus cams is displayed on the PLC interface.

These signals are irrelevant, however, or correspondingly inaccurate with the **timer-controlled** cam output variant, as described in the following paragraphs. The signals for the plus and minus cams are generated (once) in synchronism with the interpolation cycle and transmitted together to the PLC.

Pulses shorter than an interpolation cycle are thus imperceptible on the PLC.
The onboard outputs are set and reset asynchronously to the interpolation cycle for each interrupt. The status of the onboard outputs is detected and transmitted to the PLC in synchronism with the update time of the PLC interface.

Depending on the current status at the moment the PLC interface is updated, pulses shorter than one interpolation cycle are not visible or are displayed stretched by one or several IPO cycles.

Further settings

If the response described here is to be applied,
bit 0=0 must be set in the following machine data:
MD10485 SW_CAM_MODE

9.3.5 Independent, timer-controlled output of cam signals

Independent, timer-controlled cam output

Each switching edge is output separately per interrupt due to the timer-controlled, independent (of interpolation cycle) cam output.

Interaction between cam signals due to:

- single output per interpolation cycle
- output time determined by highest priority cam pair (lowest cam pair number)

does not take place.

A total of eight timer-controlled cam outputs per interpolation cycle can be configured for setting/resetting the four onboard outputs. The signal states of the plus and minus cams are also made available as standard on the PLC interface for the timer-controlled variant, but they are not relevant or accurate with a timer-controlled output.

Signal generation

MD10485 SW_CAM_MODE Bit 1
The above machine data must be set beforehand to specify
how the signals to be gated are to be generated:

Bit	Signal generation
Not set	Inversion of signal response of plus cam when: plus cam - minus cam ≥ 180 degrees
set	**No** inversion of signal response of plus cam: plus cam - minus cam ≥ 180 degrees

Settings

The assignment of cam pairs to onboard outputs
is controlled by machine data:
MD10480 SW_CAM_TIMER_FASTOUT_MASK
(screen form for output of cam signals via timer interrupts to NCU).

In addition, the general machine data below must be set to explicitly select the mode of operation:
MD10485 SW_CAM_MODE Bit0=1

Note

This function runs independently of the assignment in machine data:
MD10470 SW_CAM_ASSIGN_FASTOUT_1 or
MD10471 SW_CAM_ASSIGN_FASTOUT_2 or
MD10472 SW_CAM_ASSIGN_FASTOUT_3 or
MD10473 SW_CAM_ASSIGN_FASTOUT_4.

The onboard byte may not be used more than once at any one time.

9.4 Position-time cams

Position-time cams

The term "position-time cam" refers to a pair of software cams that can supply a pulse of a certain duration at a defined axis position.

Solution

The position is defined by a pair of software cams.
The pulse duration is defined by the lead/delay time of the plus cam.

Machine data can be used to specify that cam pairs with:
minus cam position = plus cam position
are to be processed as position-time cams.

Properties of position-time cams

- The pulse duration is independent of the axis velocity and travel direction reversal.
- The pulse duration is independent of changes in the axis position (Preset).
- Activation (rising signal edge) takes place only when the cam position is crossed.

  Moving the axis position (e.g. `PRESET`) does not trigger an activation.
- A lead/delay time is operative for the minus cam and causes a time displacement of the pulse.
- Activation (ON edge) and pulse duration are independent of the travel direction.
- The cam is not deactivated if the cam position is crossed again when the cam is active (direction reversal).
- The cam time (pulse width) is not interrupted and the cam time not restarted when the cam position is crossed again.

  This behavior is particularly relevant with respect to modulo axes, i.e. if the cam time is greater than the modulo range crossing time, the cam is not switched in every revolution.

Settings

The following settings must be made to program a position-time cam:

- **Position**

  The position must be defined by a cam pair with which the minus cam position is equal to the plus cam position.

  This is specified by setting data:
  SD41500 SW_CAM_MINUS_POS_TAB_1
  to
  SD41507 SW_CAM_PLUS_POS_TAB_4.

- **Pulse duration**

  The pulse duration is calculated by adding together the associated entries for the cam pair in:

  MD10461 SW_CAM_PLUS_LEAD_TIME[n]

  SD41521 SW_CAM_PLUS_TIME_TAB_1[n]...

  SD41527 SW_CAM_PLUS_TIME_TAB_4[n]

- **Offset**

  The time displacement of the position-time cam is calculated by adding together the associated entries for the cam pair in:

  MD10460 SW_CAM_MINUS_LEAD_TIME[n]

  SD41520 SW_CAM_MINUS_TIME_TAB_1[n]...

  SD41526 SW_CAM_MINUS_TIME_TAB_4[n]

- **Mode**

  MD10485 SW_CAM_MODE
  Bit 2 = 1 must be set in the machine data to ensure that all cam pairs with the same values for minus cam and plus cam positions are treated as position-time cams.

9.5 Supplementary conditions

Availability of function "Software cams, position switching signals"

The function is an option and is available for:

- SINUMERIK 840D with NCU 572/573, SW 2 and higher

Extensions

- The extension:
  32 instead of 16 cam pairs
  is available with software version 4.1 and higher.
- The extensions:
  independent, timer-controlled output
  position-time cams
  suppression of signal inversion for timer-controlled outputs
  are available with software version 6.3 and higher.

9.6 Data lists

9.6.1 Machine data

9.6.1.1 General machine data

Number	Identifier: $MN_	Description
10260	CONVERT_SCALING_SYSTEM	Basic system switchover active
10270	POS_TAB_SCALING_SYSTEM	System of measurement of position tables
10450	SW_CAM_ASSIGN_TAB[n]	Assignment of software cams to machine axes
10460	SW_CAM_MINUS_LEAD_TIME[n]	Lead or delay time on minus cams 1 -16
10461	SW_CAM_PLUS_LEAD_TIME[n]	Lead or delay time on plus cams 1 -16
10470	SW_CAM_ASSIGN_FASTOUT_1	Hardware assignment for output of cams 1 -8 to NCK I/Os
10471	SW_CAM_ASSIGN_FASTOUT_2	Hardware assignment for output of cams 9 -16 to NCK I/Os
10472	SW_CAM_ASSIGN_FASTOUT_3	Hardware assignment for output of cams 17 -24 to NCK I/Os
10473	SW_CAM_ASSIGN_FASTOUT_4	Hardware assignment for output of cams 25 -32 to NCK I/Os
10480	SW_CAM_TIMER_FASTOUT_MASK	Screen form for output of cam signals via timer interrupts to NCU
10485	SW_CAM_MODE	Response of SW cams

9.6.2 Setting data

9.6.2.1 General setting data

Number	Identifier: $SN_	Description
41500	SW_CAM_MINUS_POS_TAB_1[n]	Position of minus cams 1 -8
41501	SW_CAM_PLUS_POS_TAB_1[n]	Position of plus cams 1 -8
41502	SW_CAM_MINUS_POS_TAB_2[n]	Position of minus cams 9 -16
41503	SW_CAM_PLUS_POS_TAB_2[n]	Position of plus cams 9 -16
41504	SW_CAM_MINUS_POS_TAB_3[n]	Position of minus cams 17 -24
41505	SW_CAM_PLUS_POS_TAB_3[n]	Position of plus cams 17 -24
41506	SW_CAM_MINUS_POS_TAB_4[n]	Position of minus cams 25 -32
41507	SW_CAM_PLUS_POS_TAB_4[n]	Position of plus cams 25 -32

Number	Identifier: $SN_	Description
41520	SW_CAM_MINUS_TIME_TAB_1[n]	Lead or delay time on minus cams 1 -8
41521	SW_CAM_PLUS_TIME_TAB_1[n]	Lead or delay time on plus cams 1 -8
41522	SW_CAM_MINUS_TIME_TAB_2[n]	Lead or delay time on minus cams 9 -16
41523	SW_CAM_PLUS_TIME_TAB_2[n]	Lead or delay time on plus cams 9 -16
41524	SW_CAM_MINUS_TIME_TAB_3[n]	Lead or delay time on minus cams 17 -24
41525	SW_CAM_PLUS_TIME_TAB_3[n]	Lead or delay time on plus cams 17 -24
41526	SW_CAM_MINUS_TIME_TAB_4[n]	Lead or delay time on minus cams 25 -32
41527	SW_CAM_PLUS_TIME_TAB_4[n]	Lead or delay time on plus cams 25 -32

9.6.3 Signals

9.6.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Cam activation	DB31, ...DBX2.0	-

9.6.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Cams active	DB31, ...DBX62.0	-

N4: Punching and nibbling - only 840D sl

10

10.1 Brief description

Subfunctions

The functions specific to punching and nibbling operations comprise the following:

- Stroke control
- Automatic path segmentation
- Rotatable punch and die
- Clamp protection

They are activated and deactivated via language commands.

10.2 Stroke control

10.2.1 General information

Functionality

The stroke control is used in the actual machining of the workpiece. The punch is activated via an NC output signal when the position is reached. The punching unit acknowledges its punching motion with an input signal to the NC. No axis may move within this time period. Repositioning takes place after the punching operation.

High-speed signals

"High-speed signals" are used for direct communication between the NC and punching unit. Combined with the punch, they allow a large number of holes to be punched per minute since the punch positioning times are interpreted as machining delays.

PLC signals

PLC interface signals are used for non-time-critical functions such as enabling and monitoring.

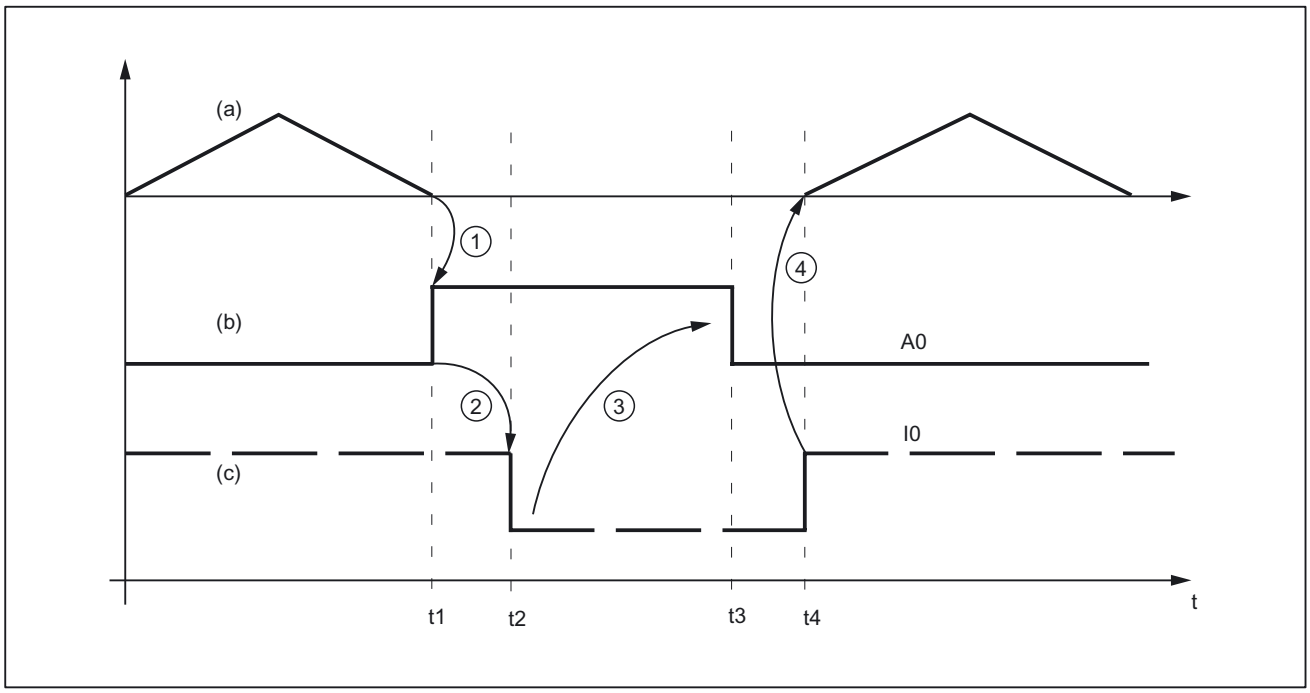
10.2.2 High-speed signals

Functionality

High-speed signals are used to synchronize the NC and punching unit. On the one hand, they are applied via a high-speed output to ensure that the punch stroke is not initiated until the metal sheet is stationary. On the other, they are applied via a high-speed input to ensure that the sheet remains stationary while the punch is active.

The high-speed digital inputs and outputs on the control are used to drive the punching unit.

The following signal chart illustrates the signal sequence.



(a) Axis motion of the machine as function v(t)
(b) "Stroke initiation" signal
(c) "Stroke active" signal

Figure 10-1 Signal chart

Note

The "Stroke active" signal is high-active for reasons relating to open-circuit monitoring.

The chronological sequence of events for punching and nibbling is controlled by the two signals A_0 and E_0:

A_0	Set by the NCK and identical to stroke initiation.
E_0	Defines the status of the punching unit and identical to the "Stroke active" signal.

The signal states characterize and define times t_1 to t_4 in the following way:

t_1	The motion of the workpiece (metal sheet) in relation to the punch is completed at instant t_1. Depending on the criterion defined for stroke initiation (refer to "Criteria for stroke initiation"), high-speed output A_0 is set for punch initiation ①.
t_2	The punching unit signals a punch movement via high-speed input E_0 at instant t_2. This is triggered by signal A_0 ②. For safety reasons, signal E_0 is high-active (in the case of an open circuit, "Stroke active" is always set and the axes do not move). The "Stroke active" signal is not reset again until the tool has moved away from the metal sheet (t_4).
t_3	The NC reacts to the "Stroke active" signal at instant t_3 by canceling the "Stroke initiation" signal ③. From this point in time onwards, the NC is in a waiting state. It simply waits for cancellation of the "Stroke active" signal so it can initiate the next axis motion. The next stroke can be initiated only after signal A_0 has disappeared.
t_4	The punching operation is complete at instant t_4, i.e. the punch has exited from the metal sheet again. The NC reacts to a signal transition in signal E_0 by starting an axis motion ④. The reaction of the NC to a signal edge change ④ is described in the section headed "Axis start after punching".

Note

The stroke time is determined by the period $\Delta t_h = t_4 - t_1$.

Reaction times at instant t_4 between the signal transition of E_0 and the start of the axis motion must also be added.

10.2.3 Criteria for stroke initiation

Initiate a stroke

The stroke initiation must be set, at the earliest, for the point in time at which it can be guaranteed that the axes have reached a standstill. This ensures that at the instant of punching, there is absolutely no relative movement between the punch and the metal sheet in the machining plane.

The following diagram shows the various criteria that can be applied to stroke initiation.

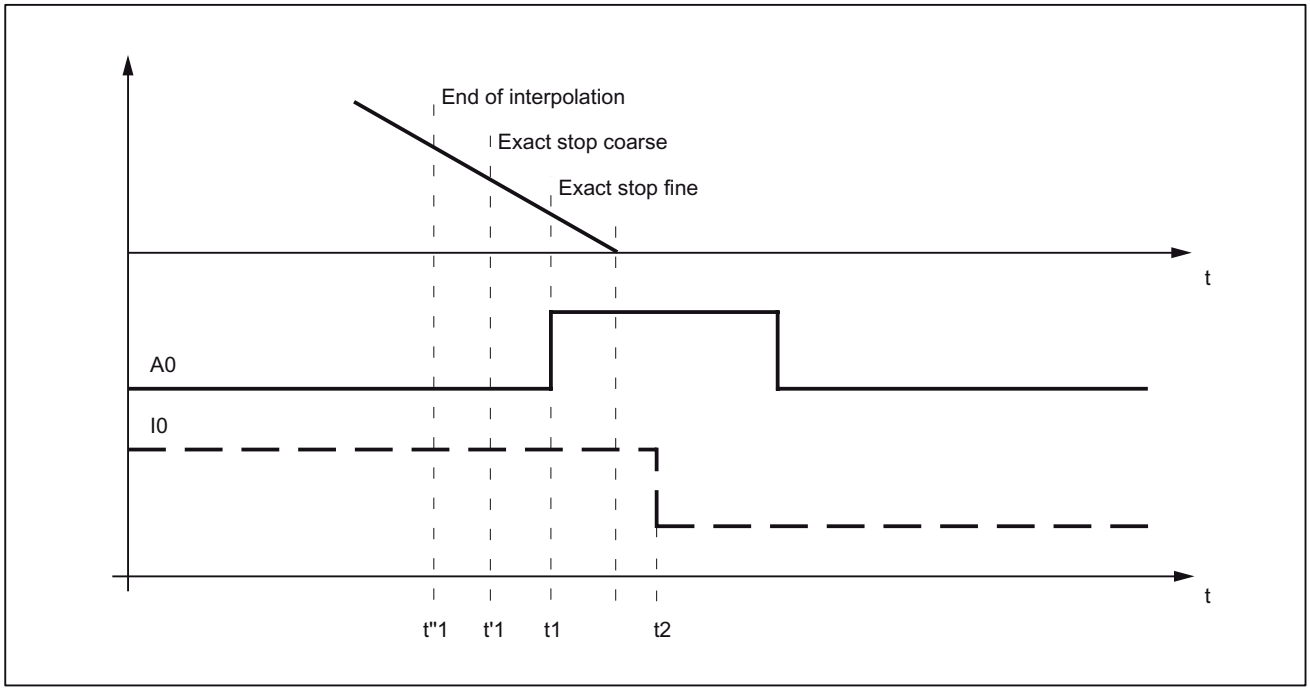


Figure 10-2 Signal chart: Criteria for stroke initiation

The time interval between t_1 and t_2 is determined by the reaction of the punching unit to setting of output A_0. This cannot be altered, but can be utilized as a lead time for minimizing dead times.

The diagram above shows the default setting with which the output is set when the "Exact stop fine window" is reached (G601; default setting of G group 12). The punch initiation times t''_1 and t'_1 are programmed by means of G602 and G603 (see table below).

Programming	Activation	Description
G603	Stop interpolation	The interpolation reaches the block end. In this case, the axes continue to move until the overtravel has been traversed, i.e. the signal is output at an appreciable interval before the axes have reached zero speed (see t''_1).
G602	Reach the coarse in-position window	The signal is output once the axes have reached the coarse in-position window. If this criterion is selected for stroke initiation output, then the instant of stroke initiation can be varied through the size of interpolation window (see t'_1).
G601	Reach the fine in-position window	In this case, it can always be ensured that the machine will have reached a standstill at the instant of punching provided that the axis data are set appropriately. However, this variant also results in a maximum dead time (see t_1).

Note

The initial setting of the G group with G601, G602 and G603 (G group 12) is defined via machine data:

MD20150 $MC_GCODE_RESET_VALUES[11]

The default setting is G601.

G603

Depending on velocity and machine dynamics, approximately 3 - 5 interpolation cycles are processed at the end of interpolation before the axes reach zero speed.

MD26018 $MC_NIBBLE_PRE_START_TIME
In conjunction with the above machine data, it is possible to delay, and therefore optimize, the instant between reaching the end of interpolation and setting the high-speed output for "Stroke ON".

The following setting data is available in addition to MD26018:

SD42402 $SC_NIBPUNCH_PRE_START_TIME

SD42402 can be changed from the part program and therefore adapted to the punching process depending on the progress of the part program run.

The following applies to the delay time:

MD26018 = 0	→ SD42402 is operative
MD26018 ≠ 0	→ MD26018 is operative

If the "Punching with dwell time, PDELAYON" is active, then the dwell time programmed in connection with this function is active. Neither MD26018 nor SD42402 is operative in this case.

10.2.4 Axis start after punching

Input signal "Stroke ON"

The start of an axis motion after stroke initiation is controlled via input signal "Stroke ON".

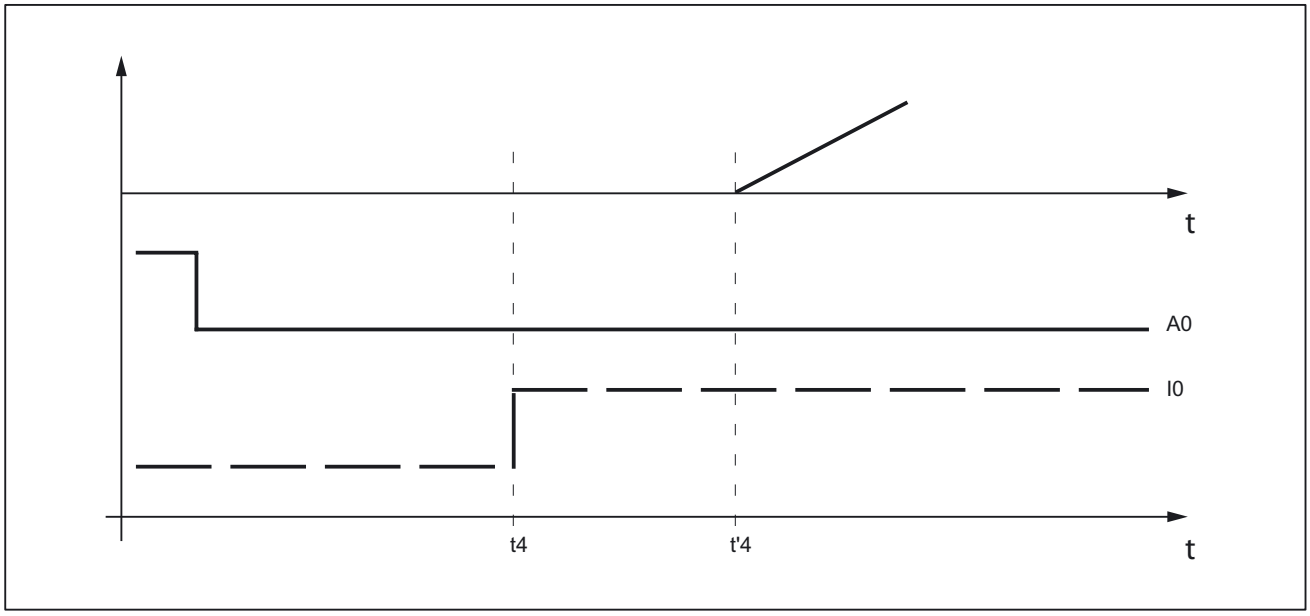


Figure 10-3 Signal chart: Axis start after punching

In this case, the time interval between t_4 and t'_4 acts as a switching-time-dependent reaction time. It is determined by the interpolation sampling time and the programmed punching/nibbling mode.

PON/SON

When the punching unit is controlled via PON/SON, the maximum delay time is calculated to be:

$|t'_4 - t_4|$ = 3 x interpolation cycles

PONS/SONS

If the punch is controlled by PONS/SONS, the delay time is determined by:

$|t'_4 - t_4| \leq$ 3 x position controller cycles

Prerequisites: Stroke time ($t_4 - t_2$) > 4 interpolation cycles

10.2.5 PLC signals specific to punching and nibbling

Function

In addition to the signals used for direct stroke control, channel-specific PLC interface signals are also available. These are used both to control the punching process and to display operational states.

Signals

Signal	Activation
DB21, ... DBX3.0 (no stroke enable)	Prevents the NC from initiating a punching operation. The NC waits until the enable signal is available before continuing the part program.
DB21, ... DBX3.2 (stroke suppression)	Allows the part program to be processed without initiating a punching operation (dry run). With active path segmentation, the axes traverse in "Stop and go" mode.
DB21, ... DBX3.4 (delayed stroke)	Activates delayed stroke output if permitted by PDELAYON.
DB21, ... DBX3.1 (manual stroke suppression)	Enables the operator to initiate a punching operation (controlled via the PLC) without executing the part program. Manual stroke initiation is acknowledged with signal: DB21, .. DBX38.1 (acknowledgement of manual stroke initiation)

10.2.6 Punching and nibbling-specific reactions to standard PLC signals

DB21, ... DBX12.3 (feed stop)

With interface signal:
DB21, ... DBX12.3 (feed stop),
the NC reacts in the following way with respect to the stroke control:

Signal is detected in advance of instant t_1:	Stroke initiation is suppressed. The next stroke is not initiated until the next start or until the "Feed stop" signal has been canceled. Machining is then continued as if there had been no interruption.
Signal is detected at instant t_1:	The current stroke is executed to completion. The NC then dwells in the state characterized by t_4. To allow it to respond in this manner, time monitoring of the "Stroke active" and "Stroke initiation" signals is dispensed with.

10.2.7 Signal monitoring

Oscillating signal

Owing to aging of the punch hydraulics, overshooting of the punch may cause the "Stroke active" signal to oscillate at the end of a stroke.

In this case, an alarm (22054 "undefined punching signal") can be generated as a function of machine data:
MD26020 $MC_NIBBLE_SIGNAL_CHECK.

Reset response

In the case of an NCK reset, the interface signal:
DB21, ... DBX38.0 (stroke initiation active)
is canceled immediately without acknowledgement by the high-speed input.

A currently activated stroke cannot be suppressed.

10.3 Activation and deactivation

10.3.1 Language commands

Punching and nibbling functions are activated and deactivated via configurable language commands. These replace the special M functions that were used in earlier systems.

References:
/PGA/ Programming Manual Job Planning

Groups

The language commands are subdivided into the following groups:

Group 35	
The actual punching and nibbling-specific functions are activated and deactivated by means of the following language commands:	
PON	= punching ON
SON	= nibbling ON
PONS	= punching ON, activated in the position controller
SONS	= nibbling ON, activated in the position controller
SPOF	= punching/nibbling OFF

Group 36	
This group includes the commands which have only a preparatory character and which determine the real nature of the punching function:	
PDELAYON	= punching with delay ON
PDELAYOF	= punching with delay OFF
Since the PLC normally needs to perform some preliminary tasks with respect to these preparatory functions, they are programmed before the activating commands.	

Group 38	
This group contains the commands for switching over to a second punch interface. It can be used, for example, for a second punching unit or set of hammer shears. A second I/O pair which can be used for punching functionality is defined via machine data.	
SPIF1	= first interface is active
SPIF2	= second interface is active

Note

Only one function at a time can be active within a G code group (similar, for example, to the various interpolation modes G0, G1, G2, G3, etc. which are also mutually exclusive).

SPOF

Punching and nibbling OFF

The SPOF function terminates all punching and nibbling functions. In this state, the NCK responds neither to the "Stroke active" signal nor to the PLC signals specific to punching and nibbling functions.

If SPOF is programmed together with a travel command in one block (and in all further blocks if punching/nibbling is not activated with SON or PON), the machine approaches the programmed position without the initiation of a punching operation. SPOF deselects SON, SONS, PON and PONS and corresponds to the Reset condition.

Programming example:

```
:
:
N20 G90 X100 SON              ;  activate nibbling
N25 X50 SPOF                  ;  deactivate nibbling,
                              ;  position without stroke initiation
:
:
```

SON

Nibbling ON

SON activates the nibbling function and deselects the other functions in G group 35 (e.g. PON).

In contrast to punching, the first stroke is made at the start point of the block with the activating command, i.e. before the first machine motion.

SON has a modal action, i.e. it remains active until either SPOF or PON is programmed or until the program end is reached.

The stroke initiation is suppressed in blocks without traversing information relating to the axes designated as punching or nibbling axes (typically those in the active plane). If a stroke still needs to be initiated, then one of the punching/nibbling axes must be programmed with a 0 traversing path. If the first block with SON is a block without traversing information of the type mentioned, then only one stroke takes place in this block since the start and end points are identical.

Programming example:

```
:
:
N70 X50 SPOF                  ;  position without punch initiation
```

```
N80 X100 SON                ;  activate nibbling, initiate a stroke before the
                            ;  motion (X=50) and on completion of the programmed
                            ;  movement (X=100)
:
:
```

SONS

Nibbling ON (in position control cycle)

SONS behaves in the same way as SON. The function is activated in the position control cycle, thus allowing time-optimized stroke initiation and an increase in the punching rate per minute.

PON

Punching ON

PON activates the punching function and deactivates SON.

PON has a modal action like SON.

In contrast to SON, however, a stroke is not executed until the end of the block or, in the case of automatic path segmentation, at the end of a path segment. PON has an identical action to SON in the case of blocks which contain no traversing information.

Programming example:

```
:
:
N100 Y30 SPOF               ;  position without punch initiation
N110 Y100 PON               ;  activate punching, punch initiation at the end of
                            ;  positioning operation (Y=100)
:
:
```

PONS

Punching ON (in position control cycle)

PONS behaves in the same way as PON. For explanation, please refer to SONS.

PDELAYON

Punching with delay ON

PDELAYON is a preparatory function. This means that PDELAYON is generally programmed before PON. The punch stroke is output with a delay when the programmed end position is reached.

The delay time in seconds is programmed in setting data:

SD42400 $SC_PUNCH_DWELLTIME

If the defined value cannot be divided as an integer into the interpolation clock cycle, then it is rounded to the next divisible integer value.

The function has a modal action.

PDELAYOF

Punching with delay OFF

PDELAYOF deactivates punching with delay function, i.e. the punching process continues normally. PDELAYON and PDELAYOF form a G code group.

Programming example:

SPIF2 activates the second punch interface, i.e. the stroke is controlled via the second pair of high-speed I/Os (see machine data MD26004 and MD26006).

```
:
:
N170 PDELAYON X100 SPOF          ;  position without punch initiation, activate
                                 ;  delayed punch initiation
:
:
N180 X800 PON                    ;  activate punching. The punch stroke is output
                                 ;  with a delay when the end position is reached.
:
:
N190 PDELAYOF X700               ;  deactivate punching with delay, normal punch
                                 ;  initiation ON. End of programmed motion
                                 ;
:
:
:
```

SPIF1

Activation of first punch interface

SPIF1 activates the first punch interface, i.e. the stroke is controlled via the first pair of high-speed I/Os (see machine data MD26004 and MD26006).

The first punch interface is always active after a reset or control system power up. If only one interface is used, then it need not be programmed.

SPIF2

Activation of second punch interface

SPIF2 activates the second punch interface, i.e. the stroke is controlled via the second pair of high-speed I/Os (see machine data MD26004 and MD26006).

Programming example:

```
:
:
N170 SPIF1 X100 PON          ;  At the end of the block, a stroke is initiated at
                             ;  the first high-speed output. The "Stroke active"
                             ;  signal is monitored at the first input.
:
:
:
:
N180 X800 SPIF2              ;  The second stroke is initiated at the second high-
                             ;  speed output. The "Stroke active" signal is
                             ;  monitored at the second input.
:
:
:
N190 SPIF1 X700              ;  All further strokes are controlled with the first
                             ;  interface.
:
```

10.3.2 Functional expansions

Alternate interface

Machines that alternately use a second punching unit or a comparable medium can be switched over to a second I/O pair.

The second I/O pair can be defined via the following machine data:

MD26004 $MC_NIBBLE_PUNCH_OUTMASK

MD26006 $MC_NIBBLE_PUNCH_INMASK

The interface is switched by command `SPIF1` or `SPIF2`.

Full punching/nibbling functionality is available on both interfaces.

Example:

Hardware assignment for stroke control

Define the high-speed byte in each case on the CPU as a high-speed punch interface:

MD26000 $MC_PUNCHNIB_ASSIGN_FASTIN = 'H00030001' → Byte 1

MD26002 $MC_PUNCHNIB_ASSIGN_FASTOUT = 'H00000001'

Remark:

The first and second bits are inverted.

Screen form for high-speed input and output bits:

MD26004 $MC_NIBBLE_PUNCH_OUTMASK[0]	= 1	First interface output bit → Bit 1 SPIF1
MD26004 $MC_NIBBLE_PUNCH_OUTMASK[1]	= 2	Second interface output bit → Bit 2 SPIF2
MD26006 $MC_NIBBLE_PUNCH_INMASK[0]	= 1	First interface input bit → Bit 1 SPIF1
MD26006 $MC_NIBBLE_PUNCH_INMASK[1]	= 2	Second interface input bit → Bit 2 SPIF2

Automatically activated pre-initiation time

Dead times due to the reaction time of the punching unit can be minimized if the stroke can be initiated before the interpolation window of the axes is reached. The reference time for this is the interpolation end. The stroke is automatically initiated with `G603` and delayed by the set value in relation to the time that the end of interpolation is reached.

The delay time for stroke initiation can be adjusted in machine data:

MD26018 $MC_NIBBLE_PRE_START_TIME

Example:

With an IPO cycle of 5 ms, a stroke shall be released two cycles after reaching the interpolation end:

⇒ MD26018 $MC_NIBBLE_PRE_START_TIME = 0.01 [s]

A pre-initiation time can also be programmed in setting data:

SD42402 $SC_NIBPUNCH_PRE_START_TIME

This setting takes effective only if MD26018 = 0 has been set.

Monitoring of the input signal

If the "stroke active" signal is fluctuating between strokes due to plunger overshoots, for example, the message "undefined punching signal" can be also be output when interpolation is stopped.

The message output is dependent on the setting in machine data:

MD26020 $MC_NIBBLE_SIGNAL_CHECK

MD26020 = 0	No alarm
MD26020 = 1	Alarm

Minimum period between two strokes

A minimum time interval between two consecutive strokes can be programmed in setting data:

SD42404 $SC_MINTIME_BETWEEN_STROKES

Example:

There must be a minimum delay of at least 1.3 seconds between two stroke initiations irrespective of physical distance:

⇒ SD42404 $SC_MINTIME_BETWEEN_STROKES = 1.3 [s]

If a punching dwell time (PDELAYON) is also programmed, then the two times are applied additively.

If a pre-initiation time at G603 is programmed, it will be effective only if the end of interpolation is reached before the time set in SD 42404:

The programmed time becomes operative immediately. Depending on the size of the block buffer, strokes that have already been programmed can be executed with this minimum interval. The following programming measures (example) can be taken to prevent this:

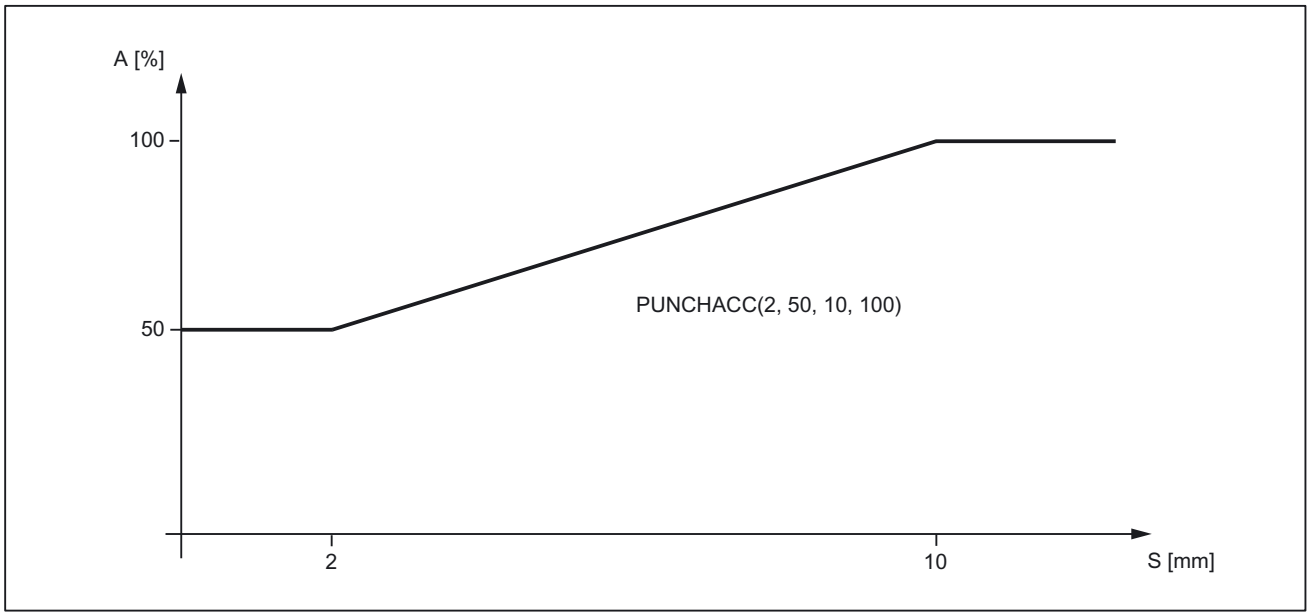
```
N...
N100 STOPRE
N110 $SC_ MINTIME_BETWEEN_STROKES = 1.3
```

The function is not active when SD42404 = 0.

Travel-dependent acceleration

An acceleration characteristic can be defined with PUNCHACC (Smin, Amin, Smax, Amax). This command can be used to define different acceleration rates depending on the distance between holes.

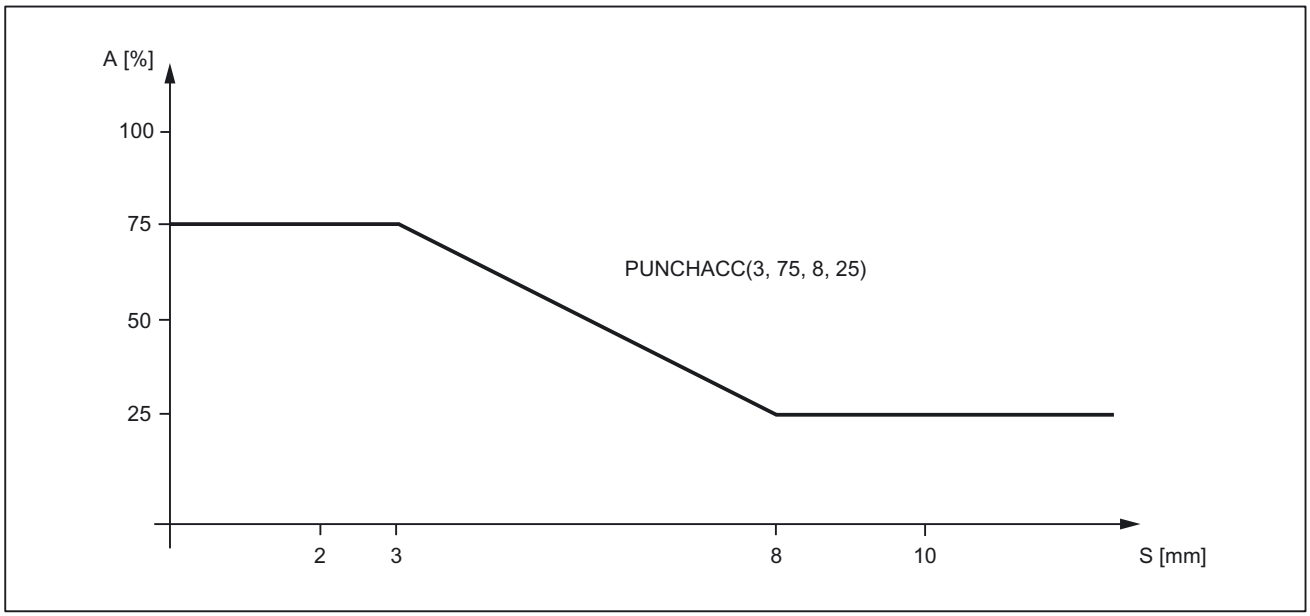
Example 1



The characteristic defines the following acceleration rates:

Distance between holes	Acceleration
< 2 mm	The axis accelerates at a rate corresponding to 50 % of maximum acceleration.
2 - 10 mm	Acceleration is increased to 100 %, proportional to the spacing.
> 10 mm (2.05 in)	The axis accelerates at the maximum rate (100%).

Example 2



The characteristic defines the following acceleration rates:

Distance between holes	Acceleration
< 3 mm	The axis accelerates at a rate corresponding to 75 % of maximum acceleration.
3 - 8 mm	Acceleration is reduced to 25 %, proportional to the spacing.
> 10 mm (2.05 in)	The axis accelerates at the maximum rate (25 %).

If a reduced acceleration rate has already been programmed via `ACC`, then the acceleration limits defined with `PUNCHACC` refer to the reduced acceleration rate.

The function is deselected with:

$S_{min} = S_{max} = 0$

The acceleration rate programmed beforehand with `ACC` remains operative.

Block search

In the case of a search for a block containing a nibbling function, it is possible to program whether the punch stroke is executed at the block beginning or suppressed.

The setting is programmed in machine data:

MD11450 $MN_SEARCH_RUN_MODE

Bit	Value	Meaning
5	0	Punch stroke at beginning of block is suppressed.
	1	Punch stroke at beginning of block is executed.

References:
/FB1/ Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1), Section "Block search"

10.3.3 Compatibility with earlier systems

Use of M functions

As in earlier versions, macro technology allows special M functions to be used instead of language commands (compatibility).

The M functions and equivalent language commands as used in earlier systems are as follows:

```
M20,M23   ≙   SPOF
M22       ≙   SON
M25       ≙   PON
M26       ≙   PDELAYON
```

Note

M functions can be configured in machine data.

When M functions are assigned to language commands, it must be noted that M functions are organized in auxiliary function groups.

Examples

```
DEFINE M20 AS SPOF
```
Punching/nibbling OFF

or

```
DEFINE M20 AS SPOF M=20
```
Punching with auxiliary function output

`DEFINE M20 AS SPOF PDELAYOF`	Punching/nibbling OFF and punching with delay OFF
`DEFINE M22 AS SON`	Nibbling ON
or	
`DEFINE M22 AS SON M=22`	Nibbling ON with auxiliary function output
`DEFINE M25 AS PON`	Punching ON
or	
`DEFINE M25 AS PON M=25`	Punching ON with auxiliary function output
`DEFINE M26 AS PDELAYON`	Punching with delay ON
or	
`DEFINE M26 AS PDELAYON M=26`	Punching and auxiliary function output

Programming example:

```
:
:
N100 X100 M20              ;  Position without punch initiation
N110 X120 M22              ;  Activate nibbling, initiate stroke before and after
                              motion
:
N120 X150 Y150 M25         ;  Activate punching, initiate stroke at end of motion
:
:
```

10.4 Automatic path segmentation

10.4.1 General information

Function

One of the following two methods can be applied to automatically segment a programmed traversing path:

- Path segmentation with maximum path segment programmed via language command SPP
- Path segmentation with a number of segments programmed via language command SPN

Both functions generate sub-blocks independently.

In earlier systems

- SPP<number> corresponds to E<number>
- SPN<number> corresponds to H<number>

Since addresses E and H now represent auxiliary functions, language commands SPP and SPN are used to avoid conflicts. The new procedure is therefore not compatible with those implemented in earlier systems. Both language commands (SPP and SPN) can be configured.

Note

The values programmed with SPP are either mm or inch settings depending on the initial setting (analogous to axes).

The automatic path segmentation function ensures that the path is divided into equidistant sections with linear and circle interpolation.

When the program is interrupted and automatic path segmentation is active (SPP/SPN), the contour can be reentered only at the beginning of the segmented block. The first punch stroke is executed at the end of this sub-block.

SPP and SPN can be activated only if geometry axes are configured.

SPP

The automatic path segmentation function SPP divides the programmed traversing path into sections of equal size according to the segment specification.

The following conditions apply:

- Path segmentation is active only when SON or PON is active.

  (Exception: MD26014 $MC_PUNCH_PATH_SPLITTING = 1)
- SPP is modally active, i.e. the programmed segment remains valid until it is programmed again, but it can be suppressed on a block-by-block (non-modal) basis by means of SPN.
- The path segments are rounded off by the control system if required so that a total programmed distance can be divided into an integral number of path sections.

- The path segment unit is either mm/stroke or inch/stroke (depending on axis settings).
- If the programmed SPP value is greater than the traversing distance, then the axis is positioned on the programmed end position without path segmentation.
- SPP = 0, reset or program end delete the programmed SPP value. The SPP value is not deleted when punching/nibbling is deactivated.

SPN

The automatic path segmentation function SPN divides the traversing path into the programmed number of path segments.

The following conditions apply:

- Path segmentation is active only when SON or PON is active.

  (Exception: MD26014 $MC_PUNCH_PATH_SPLITTING = 1)
- SPN has a non-modal action.
- Any previously programmed SPP value is suppressed for the block containing SPN, but is re-activated again in the following blocks.

Supplementary conditions

- The path segmentation function is operative with linear and circular interpolation.

  The interpolation mode remains unchanged, i.e. circles are traversed when circular interpolation is selected.
- If a block contains both SPN (number of strokes) and SPP (stroke path), then the number of blocks is activated in the current block while the stroke path is activated in all blocks that follow.
- Path segmentation is active only in conjunction with punching or nibbling functions.

  (Exception: MD26014 $MC_PUNCH_PATH_SPLITTING = 1).
- Any programmed auxiliary functions are output before, during the first or after the last sub-block.
- In the case of blocks without traversing information, the rules which govern the programming of SON and PON also apply to SPP and SPN. In other words, a stroke is initiated only if an axis motion has been programmed.

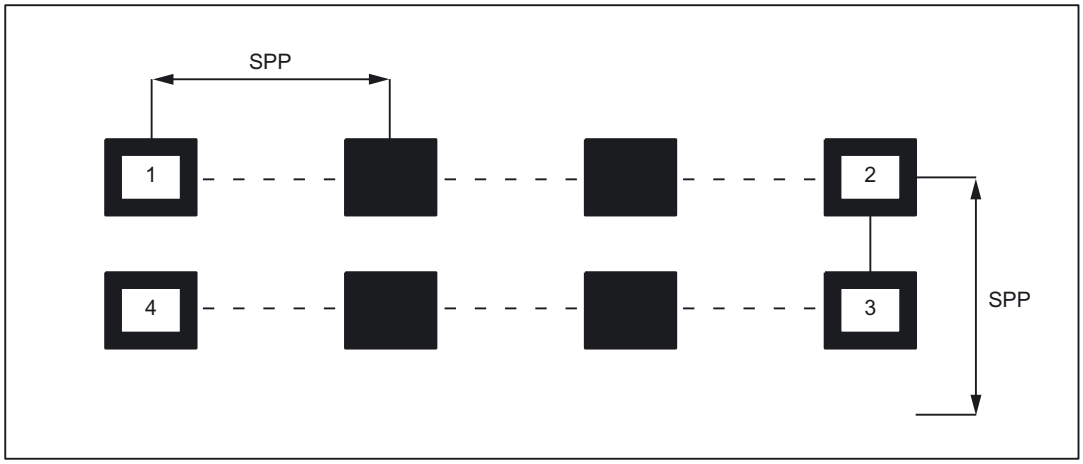
10.4.2 Operating characteristics with path axes

MD26010

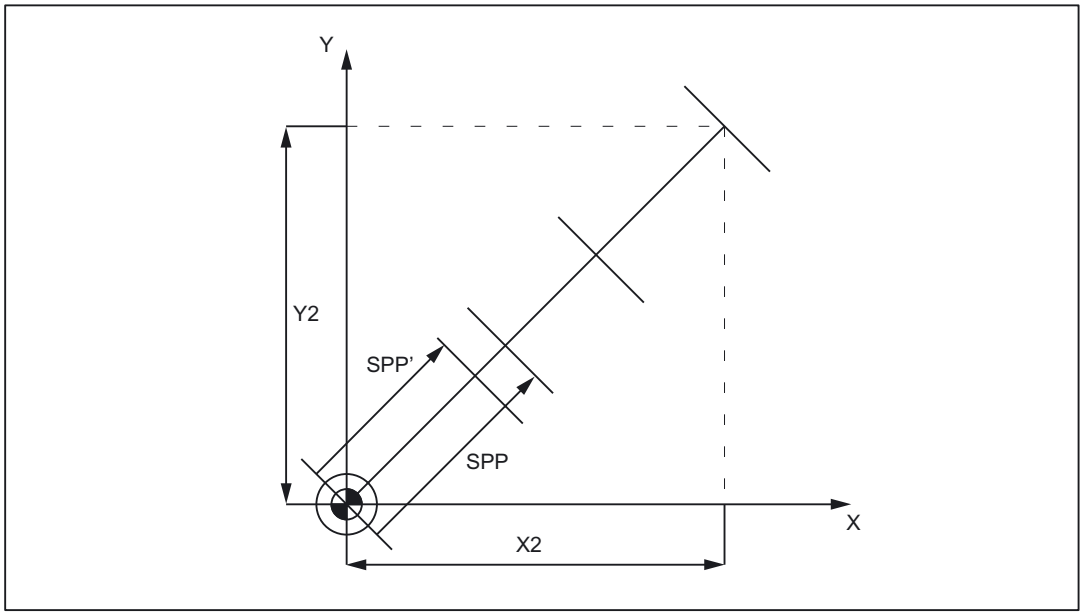
All axes defined and programmed via machine data:
MD26010 $MC_PUNCHNIB_AXIS_MASK
are traversed along path sections of identical size with SPP and SPN until the programmed end point is reached. This also applies to rotatable tool axes if programmed. The response can be adjusted for single axes.

Example of SPP

```
N1 G01 X0 Y0 SPOF          ;  Position without punch initiation
N2 X75 SPP=25 SON          ;  Nibble with feed value 25 mm; initiate punch before
                           ;  the first movement and after each path segment.
:
:
N3 Y10                     ;  Position with reduced SPP value, because travel
                           ;  distance < SPP value, and initiate punch after
                              movement.
:
:
:
N4 X0                      ;  Reposition with punch initiation after each path
                              segment.
:
```



If the programmed path segmentation is not an integral multiple of the total path, then the feed path is reduced:



X2/Y2: Programmed traversing distance
SPP: Programmed SPP value
SPP': Automatically rounded-off offset distance

Figure 10-4 Path segmentation

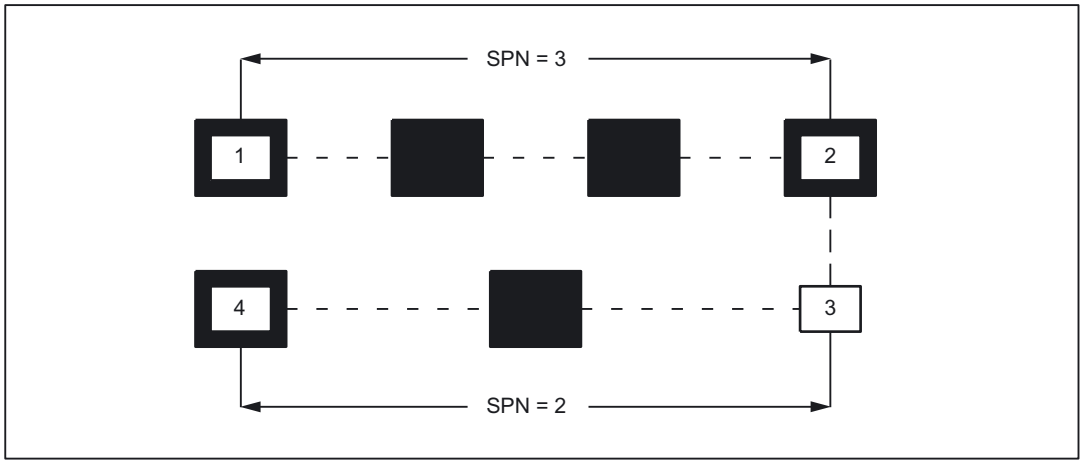
Example of SPN

The number of path segments per block is programmed via `SPN`.

A value programmed via `SPN` takes effect on a non-modal basis for both punching and nibbling applications. The only difference between the two modes is with respect to the first stroke. This is normally executed at the beginning of the first segment with nibbling operations and at its end with punching operations. This means that when n segments are programmed, n strokes are executed with punching operations but n+1 with nibbling. Furthermore, where no travel information is available, only a single stroke is executed, even if several are programmed. Should it be necessary to generate several strokes at one position, then the corresponding number of blocks without traversing information must be programmed.

```
N1 G01 X0 Y0 SPOF          ;  position without punch initiation
N2 X75 SPN=3 SON           ;  Activate nibbling. The total path is divided into
                           ;  3 segments. A stroke is initiated before the first
                           ;  movement and at the end of each segment.
:
:
:
:
```

```
N3 Y10 SPOF                ;  Position without punch initiation
N4 X0 SPN=2 PON            ;  activate punching. The total path is divided into
                           ;  2 segments. Since punching is active, the first
                           ;  stroke is initiated at the end of the first segment.
:
:
:
:
```



Example

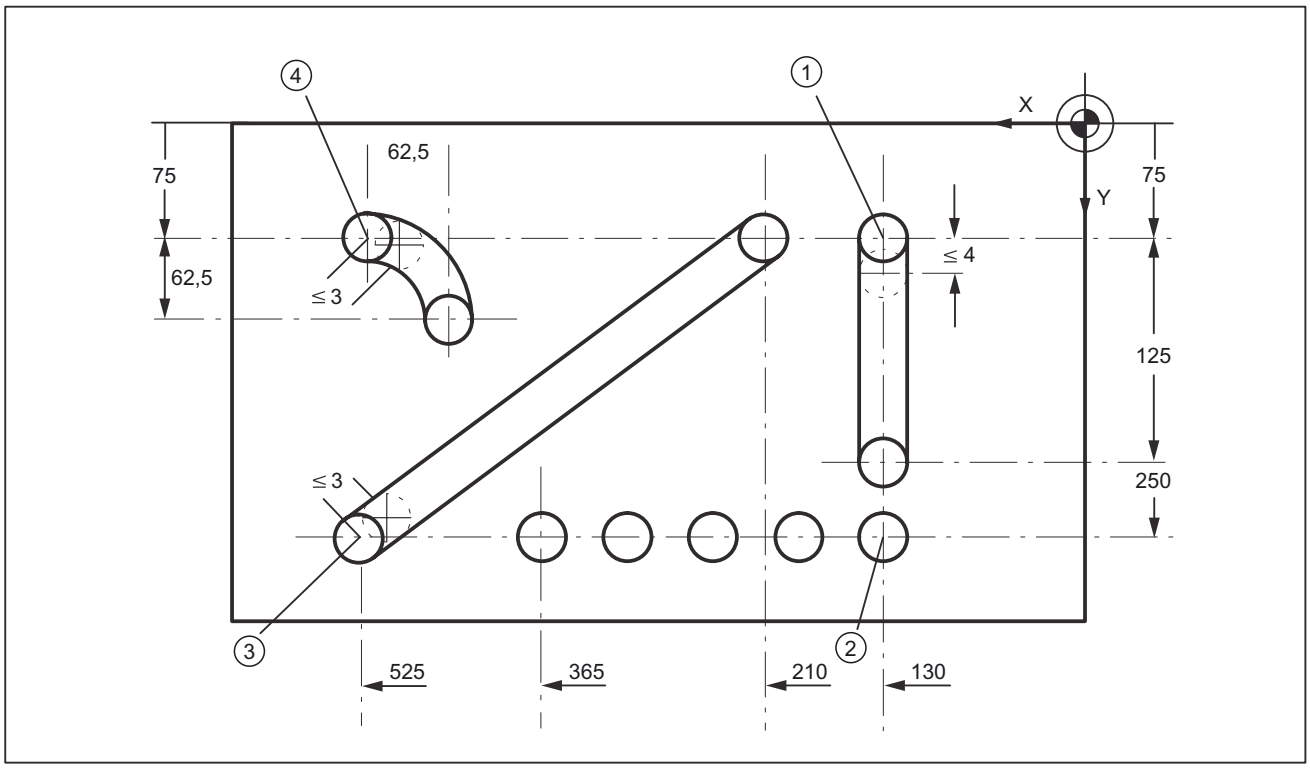


Figure 10-5 Workpiece

Extract from program

```
N100 G90 X130 Y75 F60 SPOF                ; Position at starting point □ of
                                          ; vertical nibbling path sections
N110 G91 Y125 SPP=4 SON                   ; End point coordinates (incremental);
                                          ; path segment: 4 mm, activate nibbling
N120 G90 Y250 SPOF                        ; Absolute dimensioning, position at
                                          ; starting point □ of horizontal
                                            nibbling path section
N130 X365 SPN=4 SON                       ; End point coordinates, 4 segments,
                                          ; activate nibbling
N140 X525 SPOF                            ; Position at starting point □ of
                                          ; oblique nibbling path section
N150 X210 Y75 SPP=3 SON                   ; End point coordinates path segment: 3
                                          ; mm, activate nibbling
N140 X525 SPOF                            ; Position at starting point □ of
                                          ; nibbling section on pitch circle path
N170 G02 G91 X-62.5 Y62.5 I0 J62.5 SON    ; Incremental circular interpolation
                                          ; with interpolation parameters,
                                            activate nibbling
N180 G00 G90 Y300 SPOF                    ; Position
```

10.4.3 Response in connection with single axes

MD26016

The path of single axes programmed in addition to path axes is distributed evenly among the generated intermediate blocks as standard.

In the following example, the additional rotary axis C is defined as a synchronous axis.

If this axis is programmed additionally as a "Punch-nibble axis":
MD26010 $MC_PUNCHNIB_AXIS_MASK = 1,
then the behavior of the synchronous axis can be varied as a function of machine data:
MD26016 $MC_PUNCH_PARTITION_TYPE.

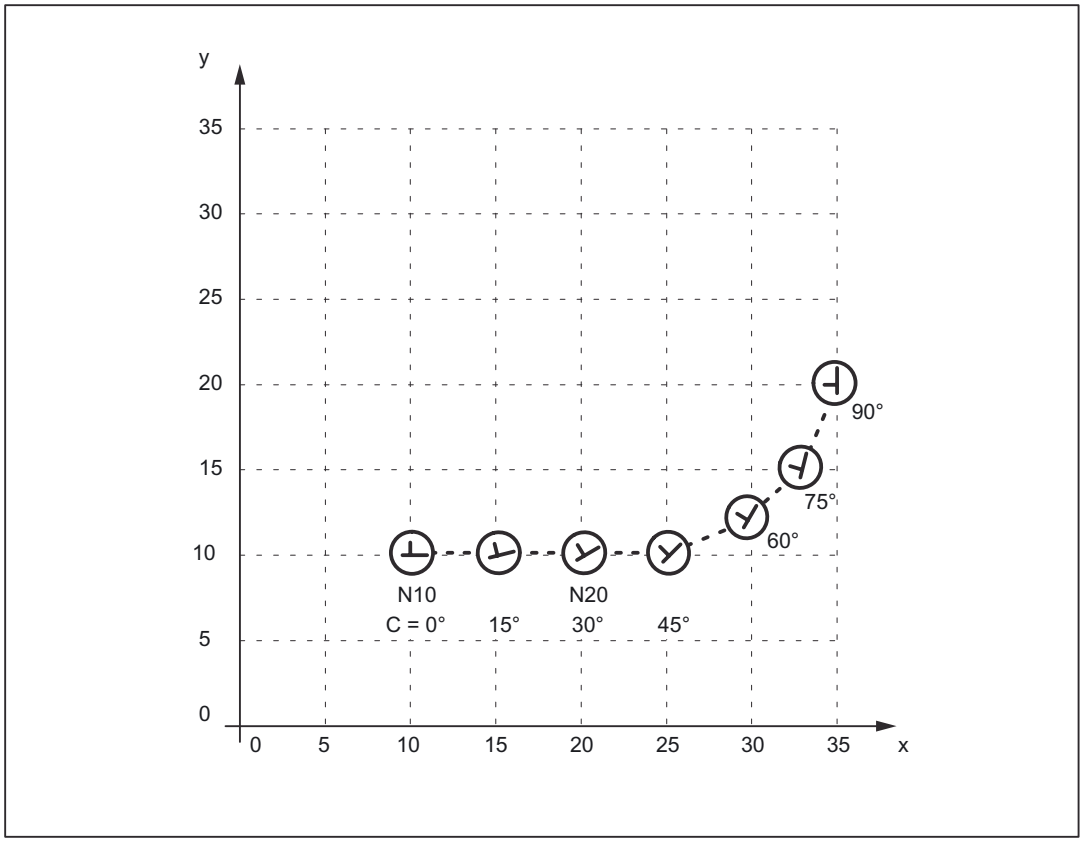
Programming example:

```
N10 G90 G1 PON X10 Y10 C0 F10000
N20 SPP=5 X25 C45
N30 G3 SPN=3 X35 Y20 I0 J10 C90
```

MD26016 $MC_PUNCH_PARTITION_TYPE=0 (default setting)

With this setting, the axes behave as standard, i.e. the programmed special axis motions are distributed among the generated intermediate blocks of the active path segmentation function in all interpolation modes.

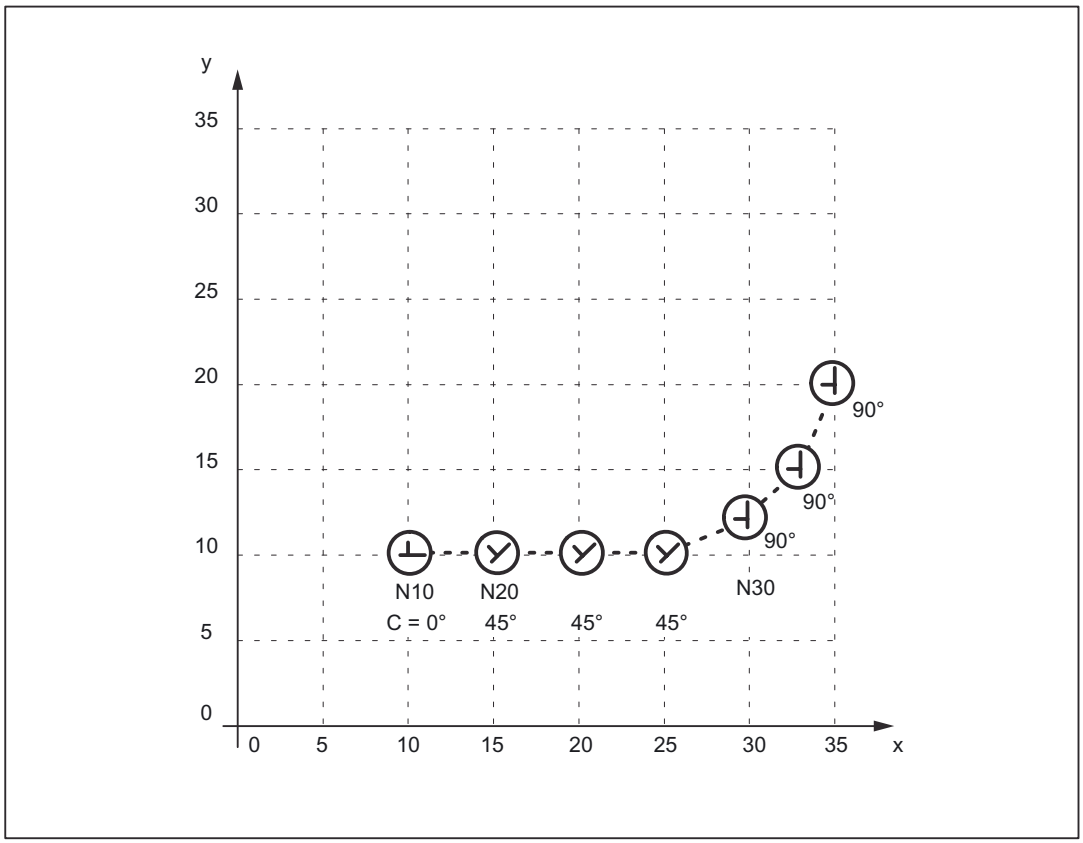
In block N20, the C axis is rotated through 15° in each of the three intermediate blocks. The axis response is the same in block N30, in the case of circular interpolation (three sub-blocks, each with 15° axis rotation).



MD26016 $MC_PUNCH_PARTITION_TYPE=1

In contrast to the behavior described above, here the synchronous axis travels the entire programmed rotation path in the first sub-block of the selected path segmentation function.

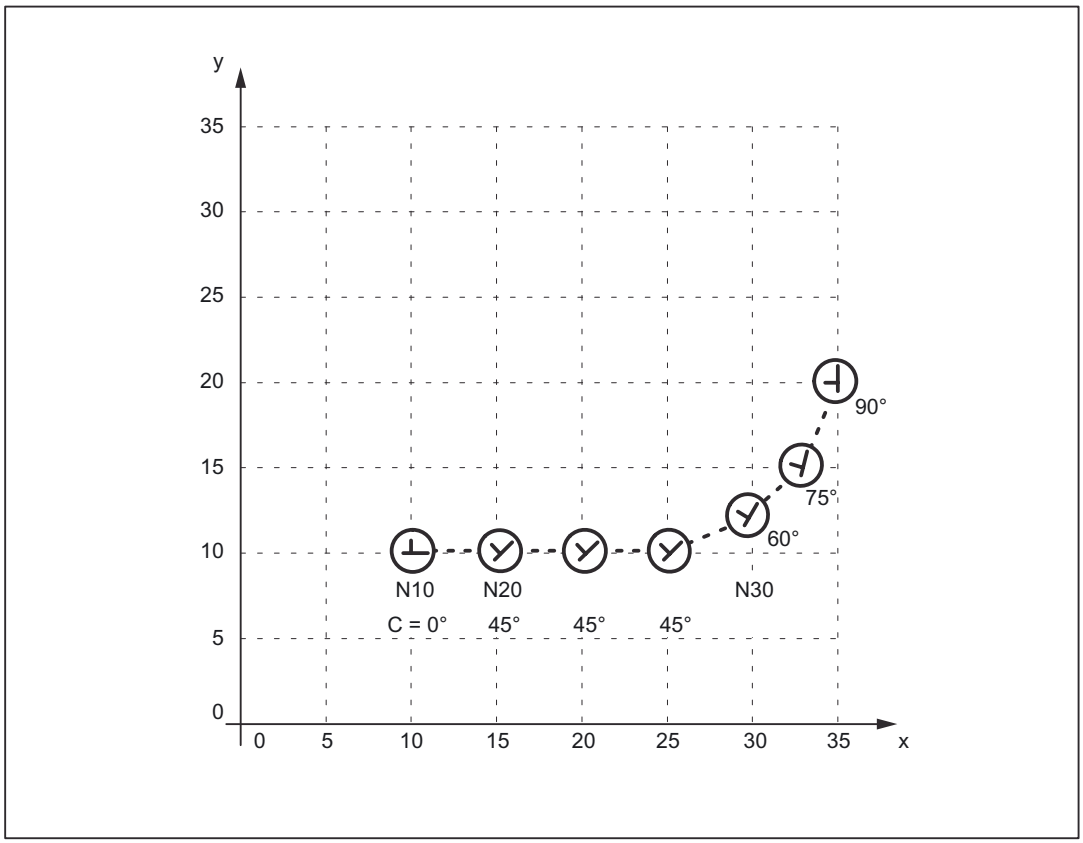
Applied to the example, the C axis already reaches the programmed end position C=45 when it reaches X position X=15. It behaves in the same way in the circular interpolation block below.



MD26016 $MC_PUNCH_PARTITION_TYPE=2

MD26016=2 is set in cases where the axis must behave as described above in linear interpolation mode, but according to the default setting in circular interpolation mode (see 1st case).

The axis behavior for the example is then as follows: In block `N20`, the C axis is rotated to C=45° in the first sub-block. The following circular interpolation block rotates the C axis through 15° in every sub-block.



The axis response illustrated in the diagram above can be particularly useful when applied to the axis of a rotatable tool in cases where it is used to place the tool in a defined direction (e.g. tangential) in relation to the contour, but where the tangential control function **must not** be applied. However, it is not a substitute for the tangential control function since the start and end positions of the rotary axis must always be programmed.

Note

Additional offset motions of special axes (in this case, rotary axis C) are implemented via a zero offset.

Supplementary conditions

- If the C axis is not defined as a "Punch-nibble axis", then the C axis motion path is not segmented in block N30 in the above example nor is a stroke initiated at the block end.
- If the functionality described above is to be implemented in a variant not specific to nibbling applications, but with alignment of the special axis, then stroke initiation can be suppressed by the following PLC interface signal:

  DB 21, 22 DBX3.2 (stroke suppression)

  (Application: e.g. alignment of electron beam during welding)

  A similar response can be programmed with the following machine data setting:

  MD26014 $MC_PUNCH_PATH_SPLITTING=1

  In this case, the path is segmented irrespective of punching or nibbling functions.

10.5 Rotatable tool

10.5.1 General information

Function overview

The following two functions are provided for nibbling/punching machines with rotatable punch and lower die:

- Coupled motion

  for synchronous rotation of punch and die

- Tangential control

  for normal alignment of rotary axes for punches in relation to workpiece

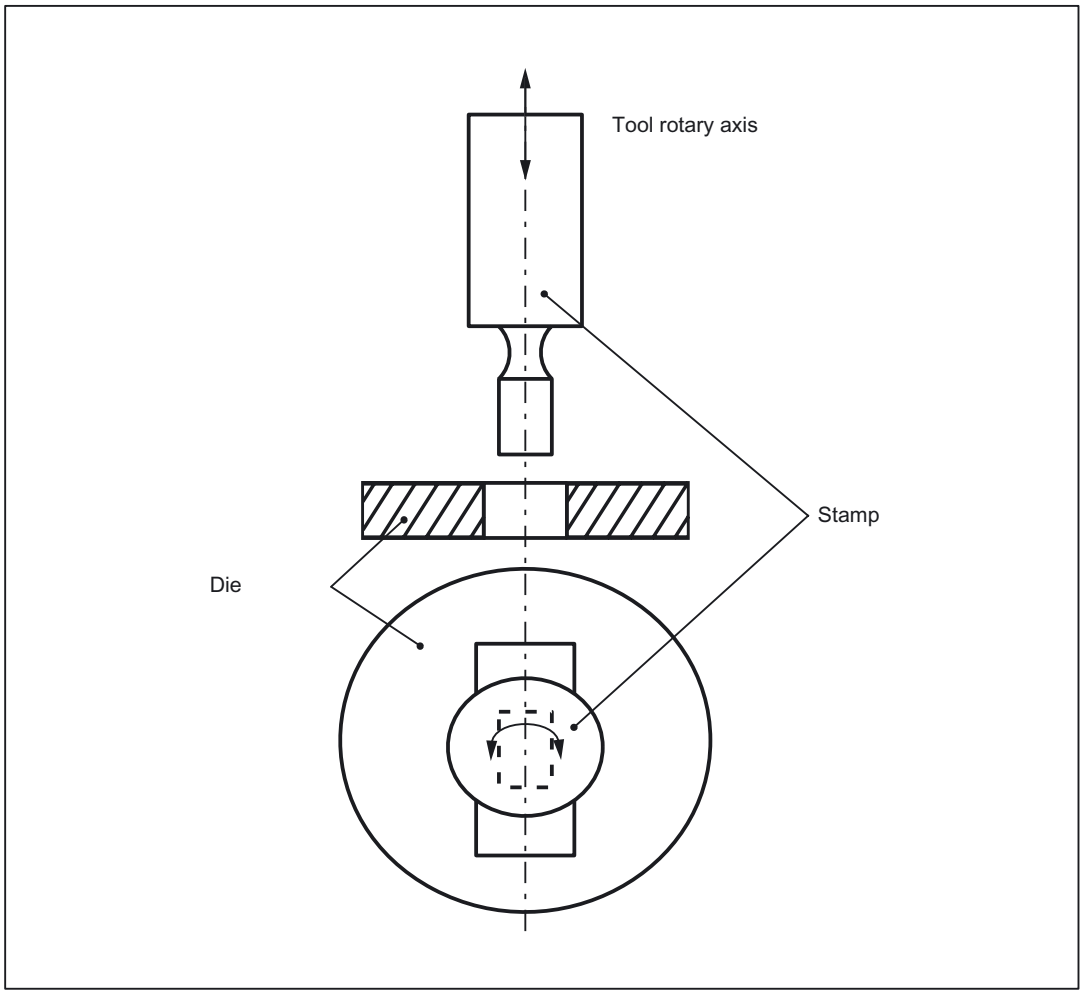


Figure 10-6 Illustration of a rotatable tool axis

10.5.2 Coupled motion of punch and die

Function

Using the standard function "Coupled motion", it is possible to assign the axis of the die as a coupled motion axis to the rotary axis of the punch.

Activation

The "Coupled motion" function is activated or deactivated with language commands `TRAILON` and `TRAILOF` respectively.

References:
/FB3/ Function Manual, Special Functions; Coupled Motion and ESR (M3)

Example

Example of a typical nibbling machine with rotatable punches where C is the punch axis and C1 the die axis:

```
:
:
TRAILON (C1, C, 1)            ;  Enabling the coupled-motion grouping
G01 X100 Y100 C0 PON          ;  Initiate stroke with C axis/C1 axis position C=0=C1
X150 C45                      ;  Initiate stroke with C axis/C1 axis position
                                 C=45=C1
:
:
M30
```

Basic position

No coupled-motion groupings are active after power up. Once the two tool axes have approached the reference point, the coupled-motion grouping is not generally separated again.

This can be achieved by:

- Program single activation of the coupled-motion grouping (see example above)
- Program MD setting:

  MD20110 $MC_RESET_MODE_MASK, bit 8=1

In this way, the coupled-motion grouping remains active after RESET/part program start or end.

10.5.3 Tangential control

Function

The rotary tool axes on punching/nibbling machines are aligned tangentially to the programmed path of the master axes by means of the "Tangential control" function.

Activation

The "Tangential control" function is activated and deactivated with language commands `TANGON` and `TANGOF` respectively.

References:
/PGA/ Programming Manual, Job Planning

Mode of operation

The tangential axis is coupled to the interpolation of the master axes. It is therefore not possible to position the axis at the appropriate punching position tangentially to the path independently of velocity. This may lead to a reduction in machining velocity if the dynamics of the rotary axis are unfavorable in relation to those of the master axes. Additional offset angles can be programmed directly via language command `TANGON`.

Note

If the tool (punch and die) is positioned by two separate drives, then the functions "Tangential control" and "Coupled motion" can be used.

Notice: The "Tangential control" function must be activated first followed by "Coupled motion".

The tangential control function automatically aligns the punch vertically to the direction vector of the programmed path. The tangential tool is positioned before the first punching operation is executed along the programmed path. The tangential angle is always referred to the positive X axis. A programmed additional angle is added to the calculated angle.

The tangential control function can be used in the linear and circular interpolation modes.

Example: Linear interpolation

The punching/nibbling machine has a rotatable punch and die with separate drives.

Programming example:

```
:
:
N2 TANG (C, X, Y, 1, "B")          ; Define master and slave axes, C is slave axis
                                     for X and Y in the base coordinate system
N5 G0 X10 Y5                       ; Start position
N8 TRAILON (C1, C, 1)              ; Activate coupled motion of rotatable tool axes
                                     C/C1
N10 Y10 C225 PON F60               ; C/C1 axis rotates to 225° → stroke
N15 X20 Y20 C45                    ; C/C1 axis rotates to 45° → stroke
N20 X50 Y20 C90 SPOF               ; C/C1 axis rotates to 90°, no stroke initiation
N25 X80 Y20 SPP=10 SON             ; Path segmentation: four strokes are executed
                                     with tool rotated to 90°
N30 X60 Y40 SPOF                   ; Position
N32 TANGON (C, 180)                ; Activate tangential control, offset angle of
                                     rotatable tool axes 180°
N35 X30 Y70 SPN=3 PON              ; Path segmentation, three strokes with active
                                     tangential control and an offset angle of 180°
N40 G91 C45 X-10 Y-10              ; C/C1 rotates to 225° (180° + 45° INC),
                                     tangential control deactivated because no path
                                     segmentation → stroke
N42 TANGON (C, 0)                  ; Tangential control without offset
N45 G90 Y30 SPN=3 SON              ; Path segmentation, three strokes with active
                                     tangential control but without offset angle
N50 SPOF TANGOF                    ; Deactivate stroke initiation + tangential
                                     control
N55 TRAILOF (C1, C)                ; Deactivate coupled motion of rotatable tool axes
                                     C/C1
N60 M2
```

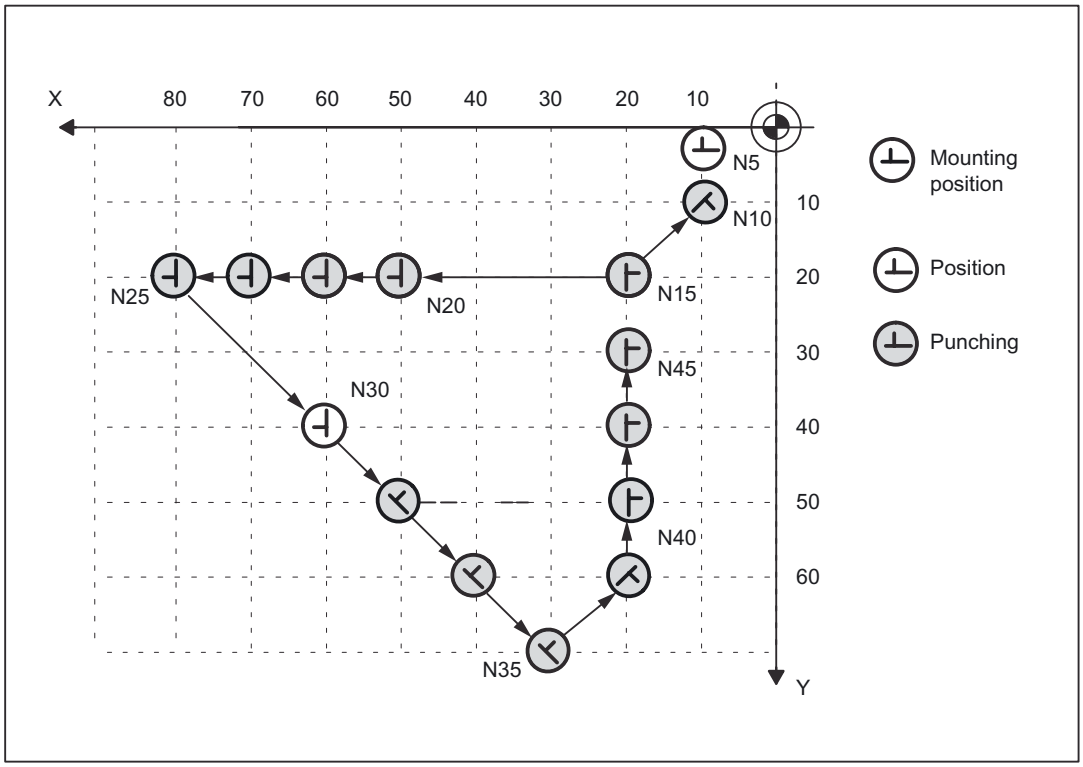


Figure 10-7 Illustration of programming example in XY plane

Example: Circular interpolation

In circular interpolation mode, particularly when path segmentation is active, the tool axes rotate along a path tangentially aligned to the programmed path axes in each sub-block.

Programming example:

```
:
:
N2 TANG (C, X, Y, 1, "B")                    ; Define master and slave axes, C
                                               is slave axis for X and Y in the
                                               base coordinate system
N5 G0 F60 X10 Y10                            ; Start position
N8 TRAILON (C1, C, 1)                        ; Activate coupled motion of
                                               rotatable tool axes C/C1 for
                                               punch and die.
N9 TANGON (C, -90)                           ; Activate tangential control with
                                               offset 270°
N10 G02 X30 Y30 I20 J0 SPN=2 PON             ; Circular interpolation with path
                                               segmentation, 2 strokes are
                                               executed with 270° offset angle
                                               and tangential alignment along
                                               circular path
N15 G0 X70 Y10 SPOF                          ; Position
```

```
N17 TANGON (C, 90)                          ; Activate tangential control with
                                              offset 90°
N20 G03 X35,86 Y24,14 CR=20 SPP=16 SON      ; Circular interpolation, path
                                              segmentation, 4 strokes are
                                              executed with 90° offset angle
                                              and tangential alignment along
                                              circular path
N25 G0 X74,14 Y35,86 C0 PON                 ; Rotation of tool axes to 0°,
                                              stroke
N27 TANGON (C, 0)                           ; Activate tangential control with
                                              offset 0°
N30 G03 X40 Y50 I-14,14 J14,14 SPN=5 SON    ; Circular interpolation, path
                                              segmentation, 5 strokes with 0°
                                              offset angle and tangential
                                              alignment along circular path
N35 G0 X30 Y65 C90 SPOF                     ; Position without active
                                              tangential control
N40 G91 X-10 Y-25 C180                      ; Positioning, C axis rotates to
                                              270°
N43 TANGOF                                  ; Deactivate tangential control
N45 G90 G02 Y60 I0 J10 SPP=2 PON            ; Circular interpolation, path
                                              segmentation, two strokes without
                                              tangential control where C=270°
N50 SPOF                                    ; Punching OFF
N55 TRAILOF (C1, C)                         ; Deactivate coupled motion of
                                              rotatable tool axes C/C1
N60 M2
```

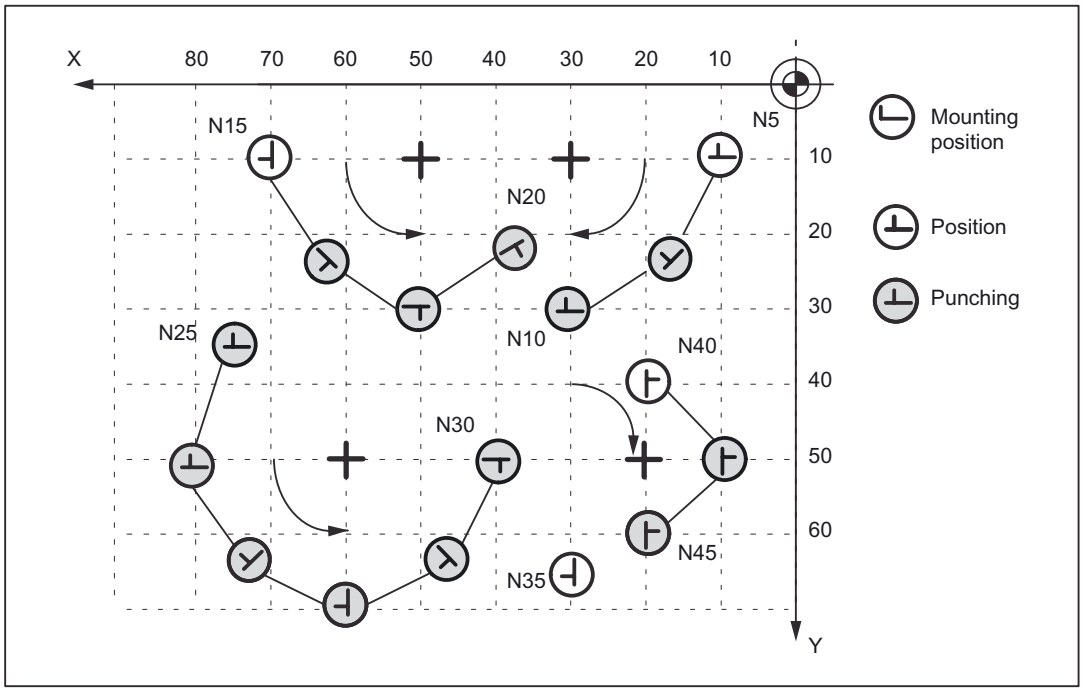


Figure 10-8 Illustration of programming example in XY plane

10.6 Protection zones

Clamping protection zone

The "clamping protection zone" function is contained as a subset in the "Protection zones" function. Its purpose is to simply monitor whether clamps and tool could represent a mutual risk.

Note

No by-pass strategies are implemented for cases where the clamp protection is violated.

References:
/FB1/ Function Manual, Basic Functions; Axis Monitoring, Protection Zones (A3)

10.7 Supplementary conditions

Availability of function "Punching and nibbling"

The function is an option and is available for:

- SINUMERIK 840D with NCU 572 and 573

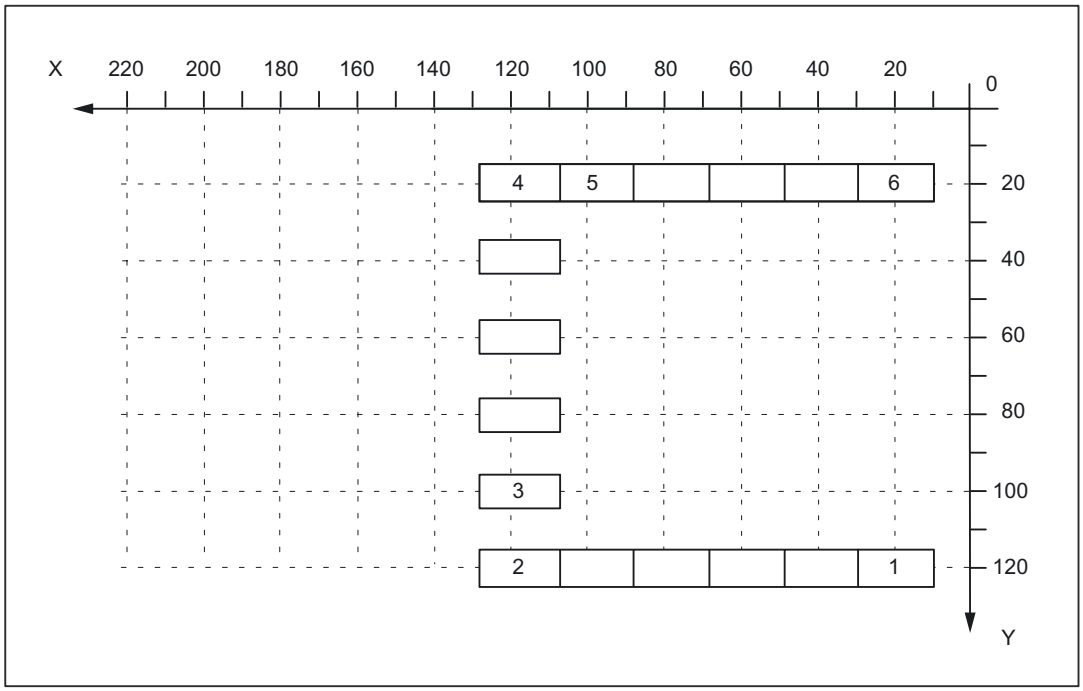
10.8 Examples

10.8.1 Examples of defined start of nibbling operation

Example 1

Example of defined start of nibbling operation

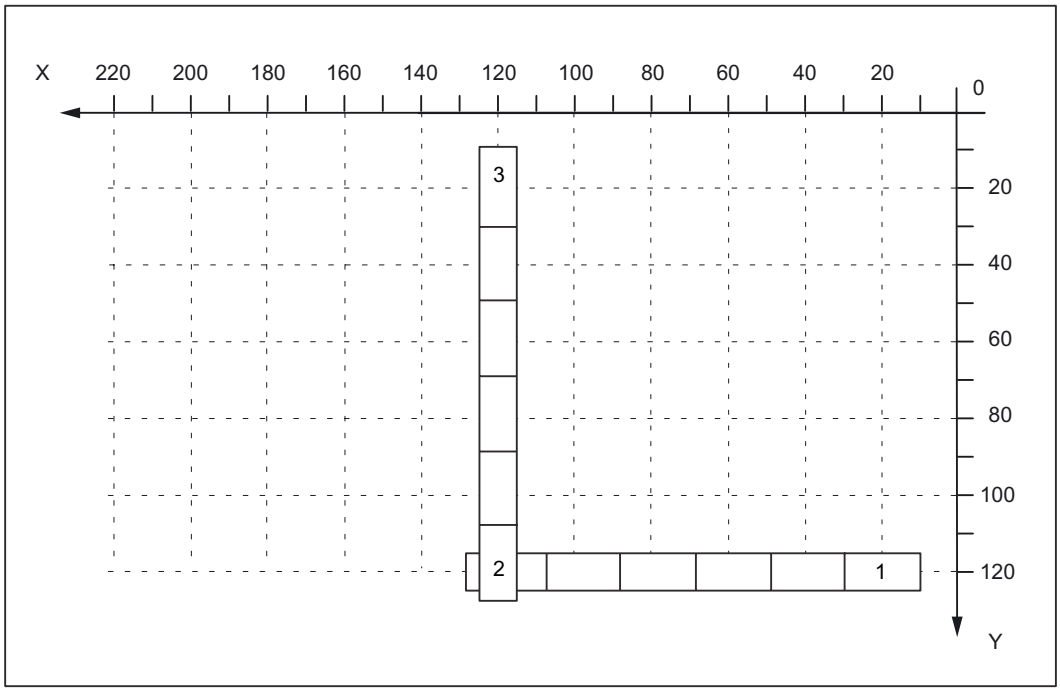
```
:
:
N10 G0 X20 Y120 SPP= 20          ;  Position 1 is approached
N20 X120 SON                     ;  Defined start of nibbling, first stroke at
                                 ;  "1", last stroke at "2"
N30 Y20                          ;  Defined start of nibbling, first stroke at
                                 ;  "3", last stroke at "4"
N40 X20                          ;  Defined start of nibbling, first stroke at
                                 ;  "5", last stroke at "6"
N50 SPOF
N60 M2
```



Example 2

This example utilizes the "Tangential control" function. Z has been selected as the name of the tangential axis.

```
:
:
N5 TANG (Z, X, Y, 1, "B")        ;  Define tangential axis
N8 TANGON (Z, 0)                 ;  Select tangential control
N10 G0 X20 Y120                  ;  Position 1 is approached
N20 X120 SPP=20 SON              ;  Defined start of nibbling, tangential control
                                    selected, first stroke at "1", last stroke at
                                    "2"
N30 SPOF TANGOF                  ;  Deselect nibbling mode and deselect tangential
                                    control
N38 TANGON (Z, 90)               ;  Select tangential control
N40 Y20 SON                      ;  Defined start of nibbling, tangential control
                                    selected, first stroke at "2" rotated 90 degrees
                                    in relation to block N20, last stroke at "3"
N50 SPOF TANGOF                  ;  Deselect nibbling mode and deselect tangential
                                    control
N60 M2
```



Examples 3 and 4 for defined start of nibbling

Example 3: Programming of SPP

```
:
:
N5 G0 X10 Y10                ;  Position
N10 X90 SPP=20 SON           ;  Defined start of nibbling, 5 punch initiations
N20 X10 Y30 SPP=0            ;  One punch is initiated at end of path
N30 X90 SPP=20               ;  4 punches initiated at intervals of 20 mm
N40 SPOF
N50 M2
```

Example 4: Programming of SPN

```
:
:
N5 G0 X10 Y10                ;  Position
N10 X90 SPN=4 SON            ;  Defined start of nibbling, 5 punch initiations
N20 X10 Y30 PON              ;  One punch is initiated at end of path
N30 X90 SPN=4                ;  4 punches initiated
N40 SPOF
N50 M2
```

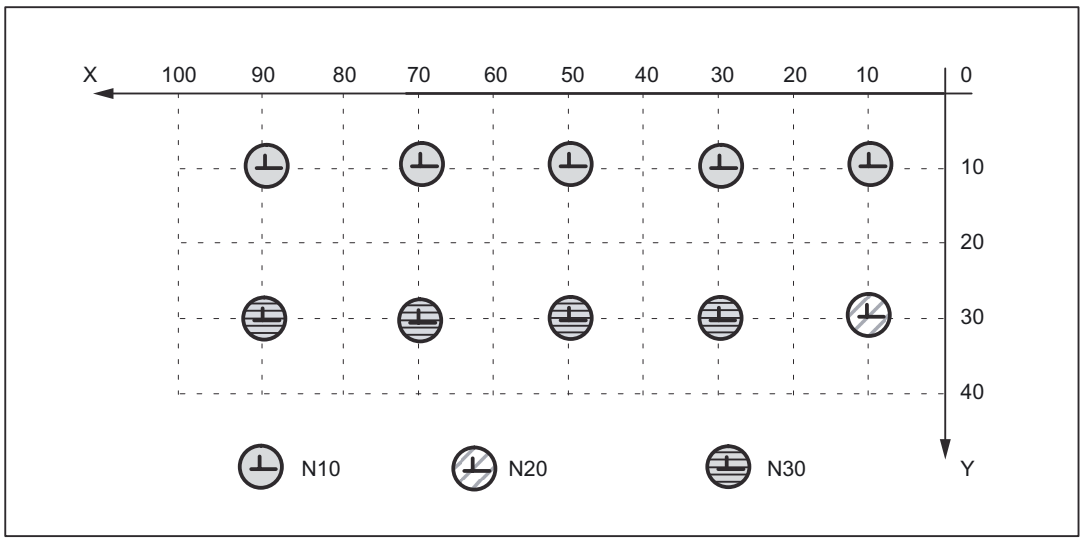


Figure 10-9 Examples 3 and 4 for defined start of nibbling

Examples 5 and 6 without defined start of nibbling

Example 5: Programming of SPP

```
:
:
N5 G0 X10 Y30                ;  Position
N10 X90 SPP=20 PON           ;  No defined start of nibbling, 4 punches initiated
N15 Y10                      ;  One punch is initiated at end of path
N20 X10 SPP=20               ;  4 punches initiated at intervals of 20 mm
N25 SPOF
N30 M2
```

Example 6: Programming of SPN

```
:
:
N5 G0 X10 Y30                ;  Position
N10 X90 SPN=4 PON            ;  No defined start of nibbling, 4 punches initiated
N15 Y10                      ;  One punch is initiated at end of path
N20 X10 SPN=4                ;  4 punches initiated
N25 SPOF
N30 M2
```

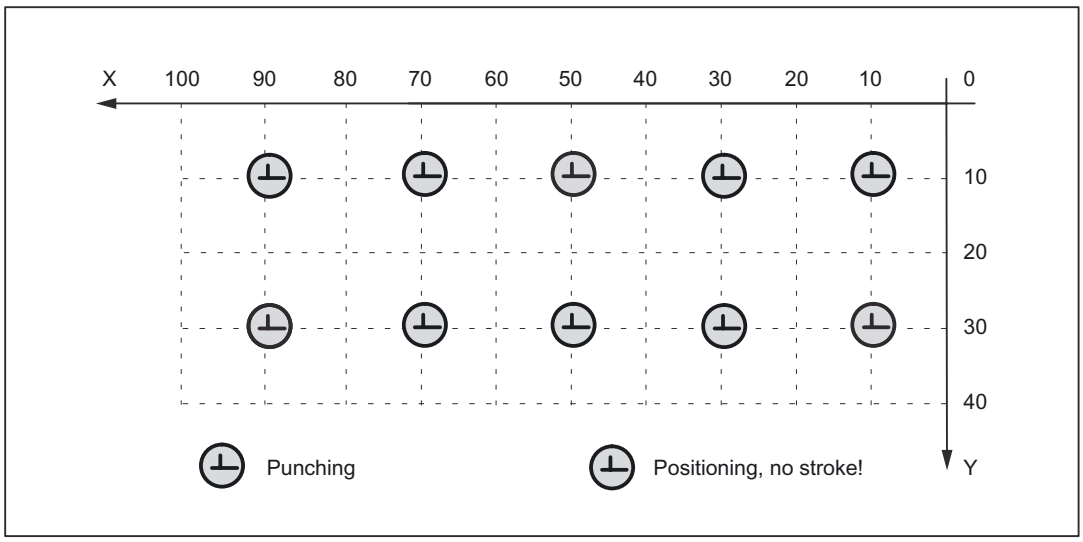


Figure 10-10 Examples 5 and 6 without defined start of nibbling

Example 7: Application example of SPP programming

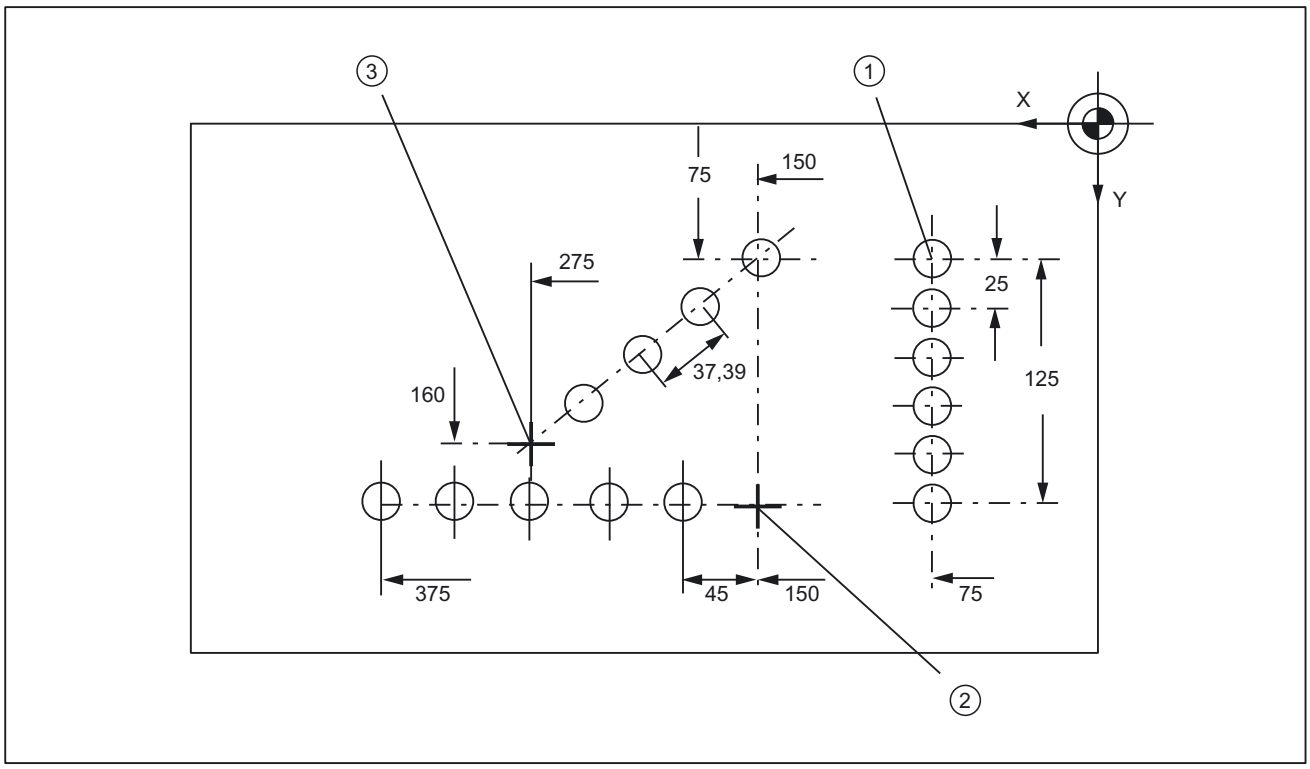


Figure 10-11 Workpiece

Extract from program:

```
N100 G90 X75 Y75 F60 PON        ;  Position at starting point 1 of vertical line of
                                   holes, punch one hole
N110 G91 Y125 SPP=25 PON        ;  End point coordinates (incremental), path
                                   segment: 25 mm, activate punching
N120 G90 X150 SPOF              ;  Absolute dimensioning, position at starting
                                   point 2 of horizontal line of holes
N130 X375 SPP=45 PON            ;  End point coordinates, path segment: 45 mm
N140 X275 Y160 SPOF             ;  Position at starting point 3 of oblique line of
                                   holes
N150 X150 Y75 SPP=40 PON        ;  End point coordinates, programmed path segment:
                                   40 mm, calculated path segment: 37.79 mm
N160 G00 Y300 SPOF              ;  Position
```

10.9 Data lists

10.9.1 Machine data

10.9.1.1 General machine data

Number	Identifier: $MN_	Description
11450	SEARCH_RUN_MODE	Block search parameter settings

10.9.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20150	GCODE_RESET_VALUES[n]	Reset G groups
26000	PUNCHNIB_ASSIGN_FASTIN	Hardware assignment for input-byte with stroke control
26002	PUNCHNIB_ASSIGN_FASTOUT	Hardware assignment for output-byte with stroke control
26004	NIBBLE_PUNCH_OUTMASK[n])	Mask for quick output bits
26006	NIBBLE_PUNCH_INMASK[n]	Mask for quick input bits
26008	NIBBLE_PUNCH_CODE[n]	Determination of the M functions
26010	PUNCHNIB_AXIS_MASK	Definition of punching and nibbling axes
26012	PUNCHNIB_ACTIVATION	Activation of punching and nibbling functions
26014	PUNCH_PATH_SPLITTING	Activation of automatic path segmentation
26016	PUNCH_PARTITION_TYPE	Behavior of single axes with active automatic path segmentation
26018	NIBBLE_PRE_START_TIME	Automatically activated pre-initiation time
26020	NIBBLE_SIGNAL_CHECK	Monitoring of the input signal

10.9.2 Setting data

10.9.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42400	PUNCH_DWELL_TIME	Dwell time
42402	NIBPUNCH_PRE_START_TIME	Pre-start time
42404	MINTIME_BETWEEN_STROKES	Minimum time interval between two consecutive strokes

10.9.3 Signals

10.9.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
No stroke enable	DB21, ...DBX3.0	-
Manual stroke initiation	DB21, ...DBX3.1	-
Stroke suppression	DB21, ...DBX3.2	-
Stroke inoperative	DB21, ...DBX3.3	-
Delayed stroke	DB21, ...DBX3.4	-
Manual stroke initiation	DB21, ...DBX3.5	-

10.9.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Stroke initiation active	DB21, ...DBX38.0	DB3300.DBX6.0
Acknowledgement of manual stroke initiation	DB21, ...DBX38.1	DB3300.DBX6.1

10.9.4 Language commands

G group	Language command	Meaning	
35	SPOF	Stroke/Punch OFF	Punching and nibbling OFF
35	SON	Stroke ON	Nibbling ON
35	SONS	Stroke ON	Nibbling ON (position controller)
35	PON	Punch ON	Punching ON
35	PONS	Punch ON	Punching ON (position controller)
36	PDELAYON	Punch with Delay ON	Punching with delay ON
36	PDELAYOF	Punch with Delay OFF	Punching with delay OFF
Path segmentation			
	SPP		Path per stroke, modal action
	SPN		Number of strokes per block, non-modal action

P2: Positioning axes

11

11.1 Brief description

Axes for auxiliary movements

In addition to axes for machining a workpiece, modern machine tools can also be equipped with axes for auxiliary movements, e.g.:

- Axis for tool magazine
- Axis for tool turret
- Axis for workpiece transport
- Axis for pallet transport
- Axis for loader (also multi-axis)
- Axis for tool changer
- Axis for sleeve assembly/end support

The axes for the workpiece machining are called path axes. Within the channel they are guided by the interpolator such that they start simultaneously, accelerate, reach the end point and stop together.

Axes for auxiliary movements are traversed independently of the path axes at a separate, axis-specific feedrate. In the past, many of these axes were moved hydraulically and started by an auxiliary function in the part program. With the closed-loop axis control implemented in the NC, the axis can be addressed by name in the part program and its actual position displayed on the screen.

Note

"Positioning axis/Auxiliary spindle" option

Axes for auxiliary movements must not be interpolating ("full-value") NC axes. Auxiliary movements may also be carried out using special axes, which can be obtained using the "Positioning axis/Auxiliary spindle" option.

Functional restrictions

Optional positioning axes/auxiliary spindles have fewer functions. The following functions are **not** possible:

- Using the axis as a path axis, geometry axis, or special path axis
- Incorporating the axis into the geometry axis grouping (GEOAX)
- Rigid thread cutting and tapping

Commissioning

As standard, axes are defined as interpolating axes:

MD30460 $MA_BASE_FUNCTION_MASK bit 8 = 0

If an axis is to be operated as a positioning axis/auxiliary spindle with reduced functionality, the value for bit 8 must be set to "1":

MD30460 $MA_BASE_FUNCTION_MASK bit 8 = **1**

Function

The "positioning axes" function makes it easier to integrate axes for auxiliary movement into the control system:

- During programming:

  The axes are programmed together with the axes for workpiece machining in the same part program, without having to sacrifice valuable machining time.

  There are special (`POS`, `POSA`) traversing instructions.

- During program testing/start-up:

  Program testing and start-up are performed simultaneously for all axes.

- During operation:

  Operation and monitoring of the machining process commence simultaneously for all axes.

- During PLC configuring/commissioning:

  No allowance has to be made on PLC or external computers (PCs) for synchronization between axes for machining and axes for auxiliary movements.

- During system configuring:

  A second channel is not required.

Motions and interpolations

Each channel has one path interpolator and at least one axis interpolator with the following interpolation functions:

- With a path interpolator:

  Linear interpolation (G1), circular interpolation (G2/G3), spline interpolation, etc.

- With an axis interpolator:

  Each channel has an axis interpolator. If a positioning axis is programmed, an axis interpolator starts in the control (with linear interpolation G1).

- End-of-motion criterion:

  The programmed end position of a positioning axis has been reached when the end-of-motion criterion FINEA, COARSA or IPOENDA is fulfilled.

- Path axes with rapid traverse movement:

  Path axes can be traversed in linear or non-linear interpolation mode with rapid traverse movement (G0).

- Autonomous single-axis operations:

  Single PLC axes, command axes started via static synchronized actions or asynchronous reciprocating axes can be interpolated independently of the NCK.

  An axis/spindle interpolated by the main run then reacts independently of the NC program. The channel response triggered by the program run is decoupled to transfer the control of a certain axis/spindle to the PLC.

- Control by PLC:

  All channel-specific signals normally act to the same extent on path and positioning axes.

  Positioning axes can be controlled via additional, axis-specific signals.

  PLC axes are traversed by the PLC via special function blocks in the basic program; their movements can be asynchronous to all other axes. The travel motions are executed separate from the path and synchronized actions.

11.2 Own channel, positioning axis or concurrent positioning axis

When axes are provided for auxiliary movements on a machine tool, the required properties will decide whether the axis is to be:

- is programmed in an internal part program (→ refer to "Own channel - only 840D sl (Page 766) ").
- programmed in the same part program as the machining operation (→ refer to "Positioning axis (posAxis) (Page 767) ").
- started exclusively by the PLC during machining (→ refer to "Concurrent positioning axis (Page 769) ").

11.2.1 Own channel - only 840D sl

A channel represents a self-contained NC which, with the aid of a part program, can be used to control the movement of axes, spindles and machine functions independently of other channels.

Non-dependence between channels

Independence between channels is assured by means of the following provisions:

- An active part program per channel
- Channel-specific interface signals such as
  - DB21, ... DBX7.1 (NC start)
  - DB21, ... DBX7.3 (NC stop)
  - DB21, ... DBX7.7 (reset)
- One feedrate override per channel
- One rapid traverse override per channel
- Channel-specific evaluation and display of alarms
- Channel-specific display, e.g. for
  - Actual axis positions
  - Active G functions
  - Active auxiliary functions
  - Current program block
- Channel-specific testing and channel-specific control of programs:
  - Single block
  - Dry run (DRY RUN)
  - Block search
  - Program test

References

For more information on the channel functionality, please refer to:
Function Manual, Basic Functions; BAG, Channel, Program Operation, Reset Response (K1)

11.2.2 Positioning axis (posAxis)

Positioning axes are programmed together with path axes, i.e. with the axes that are responsible for workpiece machining.

Instructions for both positioning axes and path axes can be included in the same NC block. Although they are programmed in the same NC block, the path and positioning axes are not interpolated together and do not reach their end point simultaneously (no direct time relationship, see also Section "Motion behavior and interpolation functions").

Positioning axis types and block change

The block change time depends on the programmed positioning axis type (refer also to Chapter "Block Change"):

Type	Description
1	The block change occurs when all path and positioning axes have reached their programmed end point.
2	The block change occurs when all path axes have reached their programmed end point. With positioning axis type 2, it is possible to approach the programmed end position across several block limits.
3	It is possible to set the block change within the braking ramp of the single axis interpolation if the criteria for the motion end and the block change are fulfilled for the path interpolation.

Motion synchronization

Positioning axes permit movements to be activated from the same machining program and such movements to be synchronized at block limits (type 1) or at explicit points by means of a `WAITP` command (type 2).

Motion end criterion for block change in the brake ramp

For single-axis interpolation, it is also possible to set another end-of-motion criterion for the block change in the braking ramp.

Traverse path axes in G0 as positioning axis

Each path axis can be traversed as positioning axis in rapid traverse movement (G0). Thus all axes travel to their endpoint independently.

In this way, two sequentially programmed X and Z axes are treated like positioning axes in conjunction with G0. The block change to axis Z can be initiated by axis X as a function of the braking ramp time setting (100-0%). Axis Z starts to move while axis X is still in motion. Both axes approach their end point independently of one another.

Axis types

Positioning axes can be linear axes and rotary axes.

Positioning axes can also be configured as indexing axes.

Independence of positioning axes and path axes

The mutual independence of path and positioning axes is ensured by the following measures:

- No shared interpolation
- Each positioning axis has a dedicated axis interpolator
- Dedicated feed override for each positioning axis
- Dedicated programmable feedrate
- Dedicated "axis-specific delete distance-to-go" interface signal.

Dependencies

Positioning axes are dependent in the following respects:

- A shared part program
- Starting of positioning axes only at block boundaries in the part program
- With rapid traverse movement G0 path axes traverse as positioning axes in one of two different modes.
- No rapid traverse override
- The following interface signals act on the entire channel and therefore on positioning axes:
  - DB21, ... DBX7.1 (NC start)
  - DB21, ... DBX7.3 (NC stop)
  - DB21, ... DBX7.7 (reset)
  - DB21, ... DBX6.1 (read-in disable)
- Alarms specific to program and channel also deactivate positioning axes.
- Program control (dry run feed, program test, DRF, ... etc.) also act on positioning axes
- Block search and single block also act on positioning axes.

- The last block with a programmed end-of-motion criterion that was processed in the search run serves as a container for setting all axes.
- Group 1 (modal movement commands) of the G functions G0, G1, G2, ...) does not apply to positioning axes.

References:
Programming Manual Fundamentals

Applications

The following are typical applications for positioning axes:

- Single-axis loaders
- Multi-axis loaders without interpolation (PTP → point-to-point traversing)
- Workpiece feed and transport

Other applications are also possible:

- With G0 workpiece delivery and workpiece transport can travel to their end points independently of one another.
- On machines with several machining processes in sequence: significant reduction in individual machining steps due to block change in the braking ramp of the single-axis interpolation.

Note

Positioning axes are not suitable for multi-axis loaders that require interpolation between the axes (path interpolator).

11.2.3 Concurrent positioning axis

Concurrent positioning axes are positioning axes with the following properties:

- Activation from the PLC need not take place at block limits, but can be implemented at any time in any operating mode (even when a part program is already being processed in the channel).
- Program command WAITP is required to move a concurrent positioning axis from the part program immediately after power ON.
- The part program continues to run uninhibited, even if the concurrent positioning axis has not reached the position defined by the PLC.
- An automatic axis change is possible, depending on the setting in the machine data MD30552 $MA_AUTO_GET_TYPE.
- With programming commands:
  - GET(<axis>) or WAITP(<axis>) becomes a concurrent positioning axis of the channel axis again.
  - "RELEASE (axis)" or WAITP(<axis>) is a channel axis that becomes a concurrent axis under PLC control.

Activation from PLC

For SINUMERIK 840D sl, the concurrent positioning axis is activated via FC 18 from the PLC.

- Feedrate

  For feedrate = 0, the feedrate is determined from the following machine data:

  MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)

- Absolute dimensions (G90), incremental dimensions (G91)

  Absolute dimensions along shortest path for rotary axes
  (<rotary axis name>=DC(<value>))

The following functions are defined:

- Linear interpolation (G1)
- Feedrate in mm/min or degrees/min (G94)
- Exact stop (G9)
- Settable zero offsets currently selected are valid

Applications

Typical applications for concurrent positioning axes include:

- Tool magazines with manual loading and unloading during machining
- Tool magazines with tool preparation during machining

11.3 Motion behavior and interpolation functions

11.3.1 Path interpolator and axis interpolator

Path interpolator

Every channel has a path interpolator for a wide range of interpolation modes such as linear interpolation (G1), circular interpolation (G2/G3), spline interpolation etc.

Axis interpolator

Each channel has axis interpolators in addition to path interpolators. The maximum number corresponds to the maximum number of existing channel axes.

If a positioning axis is programmed, an axis interpolator starts in the control with straight line interpolation G1. This axis interpolator runs independently of the path interpolator until the programmed end position of the positioning axis has been reached.

There is no time relationship between the path interpolator and the axis interpolator, nor between the axis interpolators.

Path control mode (G64) is not possible with positioning axes.

The programmed end position of a positioning axis has been reached when the end-of-motion criterion FINEA, COARSA or IPOENDA is fulfilled.

11.3.2 Interpolation response of path axis in G0

Path axes can be traversed in linear or non-linear interpolation mode in rapid traverse movement (G0).

Linear interpolation

Features:

- The path axes are interpolated together.
- The tool movement programmed with G0 is executed at the highest possible speed (rapid traverse).
- The rapid traverse velocity is defined separately for each axis in the following machine data:

  MD32000 $MA_MAX_AX_VELO
- If the rapid traverse movement is executed simultaneously on several axes, the rapid traverse speed is determined by the axis which requires the most time for its section of the path.

Linear interpolation is always performed in the following cases:

- For a G-code combination with G0 that does **not** allow positioning axis motion, e.g.:

  G40, G41, G42, G96, G961 and MD20750 $MC_ALLOW_G0_IN_G96 == FALSE
- With a combination of G0 with G64
- when a compressor or transformation is active,
- in point-to-point (PTP) travel mode
- when a contour handwheel is selected (FD=0)
- in case of an active frame with rotation of geometry axes
- if nibbling is active for geometry axes

Non-linear interpolation

Features:

- Each path axis interpolates as a single axis (positioning axis) independently of the other axes at the rapid traverse velocity defined in the following machine data:

  MD32000 $MA_MAX_AX_VELO
- The channel-specific delete distance-to-go command via the PLC and synchronized actions is applied to all positioning axes that were programmed as path axes.

In non-linear interpolation, with reference to the axial jerk:

- The setting of the concerned positioning axes BRISKA, SOFTA, DRIVEA

or

- the setting in the machine data:

  MD32420 $MA_JOG_AND_POS_JERK_ENABLE

  and

  MD32430 $MA_JOG_AND_POS_MAX_JERK

The existing system variables which refer to the distance to go ($AC_PATH, $AC_PLTBB and $AC_PLTEB) are supported.

⚠ CAUTION
As traversal of another contour is possible with non-linear interpolation, synchronized actions that refer to coordinates of the original path may not be active.

Selection of interpolation type

The interpolation type that should effective for G0 is adjusted with the following machine data:

MD20730 $MC_G0_LINEAR_MODE (interpolation response in G0)

Value	Description
0	In the rapid traversing mode (G0) the **non-linear** interpolation is active. Path axes are traversed as positioning axes.
1	In the rapid traversing mode (G0) the **linear** interpolation is active. The path axes are interpolated together.

The desired interpolation response in G0 can also be programmed via the two following part program commands, independently of the default:

RTLIOF Deactivating the linear interpolation.
⇒ In the rapid traversing mode (G0), the **non-linear** interpolation is active.

RTLION Activating the linear interpolation.
⇒ In the rapid traversing mode (G0), the **linear** interpolation is active.

The currently set interpolation response of the path axes with G0 can be queried with system variable $AA_G0MODE.

Note

In both interpolation types, rapid override is channel-specific.

11.3.3 Autonomous single-axis operations

Functionality

Single PLC axes, command axes started via static synchronized actions or asynchronous reciprocating axes can be interpolated independently of the NCK. An axis/spindle interpolated by the main run then reacts independently of the NC program with respect to:

- NC STOP
- Alarm handling
- Program control
- End of program
- RESET

Marginal conditions

Axes/spindles currently operating according to the NC program are not controlled by the PLC.

Command axis movements **cannot** be started via non-modal or modal synchronized actions for PLC-controlled axes/spindles. Alarm 20143 is signaled.

Sequence coordinator

The sequence of autonomous single-axis functions with the respective transfers is represented in a so-called **"Use Case"** overview:

NCK controls: PLC wants to accept control of the axis/spindle

Use Case 1: Cancel operation of axis/spindle

Use Case 2: Stop axis/spindle

Use Case 3: Resume axis/spindle motion

Use Case 4: Reset axis/spindle

PLC controls: PLC wants to relinquish control of the axis/spindle

Axis control by PLC

Accept control of axis/spindle

The channel behavior triggered by the NC program run is decoupled. Control of the axis/spindle is transferred via the axial VDI interface with the MC/PLC-interface signal:

DB31, ... DBX28.7 (PLC controls axis)

Description of operational sequence:

1. PLC to NCK with NST:

   DB31, ... DBX28.7 (PLC controls axis) == 1

   -> Accept control of an axis
2. NCK checks: Is the axis a main run axis or a neutral axis
3. NCK checks whether other axes can be controlled by the PLC.
4. NCK confirms acceptance and transfers the axis status with NST:

   DB31, ... DBX63.1 ("PLC controls axis") == 1 to the PLC

   System variable $AA_SNGLAX_STAT as scannable current axis status.
5. PLC has accepted control of the axis/spindle.

Alternatives

The channel status is "interrupted" because a channel stop signal is active. The axis is handled analogously to the sequence description. The following two alternatives are possible depending on the status of the axis to be controlled:

- The axis to be controlled by the PLC is **not** active.

  The Stop status is also canceled. A subsequent axis start command results directly in an axis motion.

- The axis to be controlled by the PLC is active.

  The Stop status is not canceled. Generate the axis status according to use case 2 "Stop axis". Resume the axis motion according to use case 3 "Continue axis motion".

- The channel performs an NC `RESET`.

  This process is asynchronous to control acceptance by the PLC. The two previously mentioned alternatives can occur or the axis is assigned to the channel and is reset.

Marginal conditions

The NCK must have booted.

Axes or spindles which are currently operating under the control of the NC program can be controlled by the PLC, i.e. the axis must be started by means of static synchronized actions or reciprocating axis as a neutral axis, concurrent positioning axis or command axis.

The NCK does not confirm acceptance of a program axis and outputs alarm 26072 "Channel %1 axis %2 cannot be controlled by PLC".

If the value is overwritten by the value set in the options data, then a transfer to the PLC is not possible.

Alarm 26070 "Channel %1 axis %2 cannot be controlled by PLC, max. no. exceeded" is generated.

Relinquish axis control by PLC

Relinquish control of axis/spindle

Control of the axis/spindle is transferred via the axial VDI interface with the MC/PLC-interface signal:

DB31, ... DBX28.7 (PLC controls axis)

Description of operational sequence:

1. PLC to NCK with NST:

   DB31, ... DBX28.7 (PLC controls axis) == 0

   -> Relinquish control of an axis.

2. NCK checks, whether an axial alarm is active
3. NCK checks whether a movement has been activated that is not yet ended and stops this movement with an axial stop as per use case 2 "Stop axis/spindle".

4. NCK executes an axial RESET in accordance with use case 4 "Reset axis/spindle" by reading and activating the required reset machine data for RESET for a single axis.
5. NCK confirms the takeover and transfers the axis status to the PLC via the axial VDI interface with the NC/PLC-interface signals:

   DB31, ... DBX63.1 (PLC controls axis) == 0

   DB31, ... DBX63.2 (Axis stop active) == 0

   DB31, ... DBX63.0 (AXRESET DONE) == 0

   Axis status active with system variable $AA_SNGLAX_STAT == 0.
6. The channel now has control over the axis/spindle.

Alternatives

The NCK confirms the transfer and internally sets the "stopped" channel status for the axis/spindle if it detects that:

- the axis-control channel is in the "interrupted" status owing to a stop signal,
- a stop alarm is active for the channel or
- a stop alarm is active for the mode group.

Marginal conditions

The axis/spindle must be operating under PLC control.

The NCK confirms acceptance of an axis/spindle only if an axial alarm is not active. If an axial alarm is active, alarm 26074 "Channel %1 Deactivation of PLC control of axis %2 in current status not permitted" is generated.

Applications

An axis/spindle interpolated by the main run can be controlled by the PLC independently of the NC program via the following VDI interface signals:

- Cancel operation:

  DB21, ... DBX6.2 (delete distance-to-go)
- Stop axis/spindle:

  DB31, ... DBX28.6 (AxStop, stop)

- Resume axis/spindle:

  DB31, ... DBX28.2 (AXRESUME)

- Reset axis/spindle:

  DB31, ... DBX28.1 (AXRESET)

Note

The axis/spindle must be operating under PLC control. This supplementary condition basically applies to all applications: Use cases 1 to 4.

The exchange of signals at the VDI interface during autonomous single operations is described by means of machine axis 1 in a comparison of PLC actions as the NCK reaction in Section "Control by the PLC".

For further information about channel-specific VDI signals, please refer to:

References
/FB1/ Function Manual, Basic Functions; Various Interface Signals (A2)

Alarm handling for single axes

Alarms with an axis parameter are only displayed and not handled as single axis alarms. It is sufficient to acknowledge these so-called "show alarms" with `CANCEL`.

Note

Only alarms that have to be cleared again with AXRESET take effect on single axes.

Use Case 1

Cancel operation of axis/spindle

The function Cancel axis/spindle corresponds to the current behavior of the channel-specific NC/PLC-interface signals:

DB21, ... DBX6.2 (delete distance-to-go)

Use Case 2

Stop axis/spindle

All axis motions controlled by the main run are stopped, thereby interrupting the following movements:

- Axis/spindle:

  PLC axis, asynchronous reciprocating axis or command axis started by static synchronized action. Following axis motion of the axis/spindle is not stopped.

- Overlaid movement such as:

  $AA_OFF,DRF handwheel traversal or online tool offset and external zero offset.

Description of operational sequence:

- PLC requests the NCK to stop the relevant axis with NST:
  DB31, ... DBX28.6 ("AxStop, stop") == 1.
- NCK brakes the axis along a ramp.
- NCK switches the axis to the 'stopped' state and notifies the PLC of the status change via the VDI interface (NCK→PLC) as follows with:

  DB31, ... DBX63.2 (Axis stop active) == 0,

  DB31, ... DBX64.7 ("Motion command plus") == 0 or

  DB31, ... DBX64.6 ("Motion command minus") == 0 and

  DB31, ... DBX60.7 (Exact stop fine) == 1 or

  DB31, ... DBX60.6 (Exact stop coarse) == 1

  Axis status interrupted with system variable $AA_SNGLAX_STAT == 3
- NCK ends this operation.

Alternatives

The following scenarios for stopping the axis/spindle are possible:

- Accept and stop the axis/spindle simultaneously.

  If PLC notifies the NCK: PLC wants to accept the axis/spindle and it is simultaneously detected that the axis/spindle is to be stopped, then: NCK flags the axis/spindle as accepted and continues at the position where the interruption occurred.
- The axis/spindle is executing several operating sequences:

  Stop all operations except for movements resulting from following axis motions. Following axis motion can only be stopped by stopping the leading axis!
- PLC requests the NCK to stop the axis/spindle, an axial stop alarm for this axis is generated simultaneously:

  The NCK brakes the axis along a ramp and confirms the braking operation via the OPI.

  At the same time the PLC alarm is sent with:
  DB31, ... DBX61.1 ("Axial alarm") == 1 and
  the status system variable $AA_SNGLAX_STAT == 5 is set.

  The NCK switches the axis to the Stopped state and signals this status change to the PLC.
- The PLC requests the NCK to stop the axis/spindle, a stop alarm is generated simultaneously and the NC program is activated. The stop alarm is ignored and does not take effect.
- A retraction movement of "Extended stop and retract" cannot be stopped. This retraction motion cannot be halted with a NST:
  DB31, ... DBX28.6 (AxStop, Stop)
  on the PLC interface.

Marginal conditions

The PLC must already have accepted control of the axis/spindle. Else, the following NC/PLC-interface signal is ignored:

NST DB31, ... DBX28.6 (AxStop, stop)

Use Case 3

Resume axis/spindle motion

The axis/spindle motions controlled by the main run and interrupted according to use case 2 "Stop axis" are resumed.

Description of operational sequence:

- PLC requests the NCK to resume motion on the relevant axis with NST: DB31, ... DBX28.2 (AXRESUME) == 1.
- NCK checks whether an axis/spindle axial alarm with the cancel criterion CANCELCLEAR or NCSTARTCLEAR is active and deletes it.
- NCK checks whether the axis motion can be resumed - no interlocking due to an alarm - and switches it to the active state.
- The axis motion is resumed and the PLC notified of the status change via the VDI interface (NCK→PLC) as follows:

  DB31, ... DBX63.2 (Axis stop active) == 0,

  DB31, ... DBX64.7 (Motion command plus) == 1 or

  DB31, ... DBX64.6 (Motion command minus) == 1 and

  DB31, ... DBX60.7 (Exact stop fine) == 0 or

  DB31, ... DBX60.6 (Exact stop coarse) == 0

  Axis status active with system variable $AA_SNGLAX_STAT == 4.
- NCK ends this operation.

Alternatives

The axis/spindle is executing several operating sequences:

- Resume all operating sequences.
- Following motions are dependent on the leading axis motion.

Marginal conditions

The NCK detects the following fault scenarios:

- When the PLC prompts the axes/spindles to continue moving and this axis/spindle has not been accepted by the PLC, the IS DB31, ... DBX28.2 ("AXRESUME") is ignored.
- If the axis/spindle is not in a stopped state, the NST DB31, ... DBX28.2 (AXRESUME) is ignored.
- If the motion of the axis/spindle must not be continued due to an active alarm, the NST DB31, ... DBX28.2 (AXRESUME) is ignored.

Use Case 4

Reset axis/spindle

An axis/spindle is reset to its initial state.

Description of operational sequence:

- PLC requests the NCK to reset motion on the relevant axis with NST:
  DB31, ... DBX28.1 (AXRESET) == 1.
- NCK detects that the axis/spindle is active and switches it to the stopped state.
- The interrupted sequences are canceled and notified to the PLC as for "Delete distance to go".
- The internal NCK states for the axis/spindle are reset.
- The axial machine data effective on RESET become operative.

Note

In the case of a channel reset, no axial machine data becomes operative for an axis controlled by the PLC.

- An axial RESET has been performed is reported at the interface:

  DB31, ... DBX63.0 ("AXRESET DONE") == 1

  DB31, ... DBX63.2 ("Axis stop active") == 0

  and

  System variable $AA_SNGLAX_STAT == 1 single axis in RESET active.
- NCK ends this operation.

Alternatives

The NCK detects that the relevant axis/spindle:

- is in the state "stopped":

  → continue flow description from point 3 onwards.
- is in the state "StopByAlarm":

  → continue flow description from point 3 onwards.
- is not active:

  → continue flow description from point 4 onwards.

Marginal conditions

The PLC must already have accepted control of the axis/spindle. NST DB31, ... DBX28.1 (AXRESET) is otherwise ignored.

11.3.4 Autonomous single-axis functions with NC-controlled ESR

Extended stop numerically controlled

The numerically controlled extended stop and retract function is also available for single axes and is configurable with axial machine data:

Delay time for ESR single axis with

MD37510 $MA_AX_ESR_DELAY_TIME1

ESR time for interpolatory braking of the single axis with

MD37511 $MA_AX_ESR_DELAY_TIME2

The values of these axial machine data are however effective only if the axis/spindle is a single axis.

The NC-controlled extended stop and retract is activated by the axial trigger $AA_ESR_TRIGGER[axis]. It works analogously to $AC_ESR_TRIGGER and applies exclusively to single axes.

References:

/FB3/ Function Manual, Special Functions; Coupled axes and ESR (M3)

Extended retract numerically controlled

For retracting single axes, the value must have been programmed via POLFA(axis, type, value) and the following conditions must be met:

- The axis must be a single axis at the time of triggering
- $AA_ESR_ENABLE[axis]=1
- POLFA(axis, type, value) with type=1 or type=2 only

  POLFA(axis, value, axis, type, axis type).

Note

NC-controlled extended stop for single axes:

The trigger is only effective if the axis is a single axis at the time of triggering, otherwise the trigger is ignored and the axial stop for this axis is **not** executed.

NC-controlled extended retract for single axes:

The channel-specific NC-controlled extended retract function is **not** effective for single axes. All axes that are single axes at the time of triggering $AC_ESR_TRIGGER will be ignored for channel-specific retraction.

This also applies when all the parameters for retraction are set, such as:

MD37500 $MA_ESR_REACTION

$AA_ESR_ENABLE for the axis, etc.

Examples

Extended **stopping** of a single axis:

MD37500 $MA_ESR_REACTION[AX1]=22

MD37510 $MA_AX_ESR_DELAY_TIME1[AX1]=0.3

MD37511 $MA_AX_ESR_DELAY_TIME2[AX1]=0.06

...

$AA_ESR_ENABLE[AX1] = 1

$AA_ESR_TRIGGER[AX1]=1 ; axis begins stop process here

Extended **retraction** of a single axis:

MD37500 $MA_ESR_REACTION[AX1]=21

...

$AA_ESR_ENABLE[AX1] = 1

POLFA(AX1, 1, 20.0); AX1 is assigned the axial retraction position 20.0 ; (absolute)

$AA_ESR_TRIGGER[AX1]=1 ; AX1 begins to retract here

POLFA(axis, type): permissible programming abbreviation

POLFA(axis, 0/1/2) ; quick deactivation/activation

⚠WARNING

If abbreviated notation is used and only the type is changed, make sure that the value for the retraction position or retraction path in the application is meaningful!

The abbreviated notation should only be used in exceptional circumstances.

This particularly applies after:

A power on, the retraction path or the retraction position must be reset.

POLFA(axis, 1, $AA_POLFA[axis]) ; causes a preprocessing stop

POLFA(axis, 1); does **not** cause a preprocessing stop

11.4 Velocity

The axis-specific velocity limits and acceleration limits are valid for positioning axes.

Feed override

The path and positioning axes have separate feedrate overrides. Each positioning axis can be adjusted by its own axis-specific feed override.

Rapid traverse override

Rapid traverse override applies only to path axes. Positioning axes have no rapid traverse interpolation (only axial linear interpolation G01) and therefore no rapid traverse override.

feed

The positioning axes traverse at the axis-specific feedrate programmed for them. As illustrated in section "Motion behavior and interpolation functions", the feedrate is not influenced by the path axes.

The feedrate is programmed as an axis-specific velocity in units of min/mm, inch/min or degrees/min.

The axis-specific feedrate is always permanently assigned to a positioning axis by the axis name.

If a positioning axis has no programmed feedrate, the control system automatically applies the rate set in axis-specific machine data:

MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity).

The programmed axis-specific feedrate is active until the end of the program.

Rev. feedrate

In JOG mode the behavior of the axis/spindle also depends on the setting of SD41100 JOG_REV_IS_ACTIVE (revolutional feedrate when JOG active).

- If this setting data is active, an axis/spindle is always moved with revolutional feedrate MD32050 $MA_JOG_REV_VELO (revolutional feedrate with JOG) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate with JOG with rapid traverse overlay) as a function of the master spindle.
- If the setting data is not active, the behavior of the axis/spindle depends on SD43300 $SA_ASSIG_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles)
- If the setting data is not active, the behavior of a geometry axis on which a frame with rotation is effective depends on the channel-specific setting data SD42600 $SC_JOG_FEED_PER_REV_SOURCE. (In the operating mode JOG, revolutional feedrate for geometry axes on which a frame with rotation is effective).

11.5 Programming

11.5.1 General

Note

For the programming of position axes, please observe the following documentation:
References:
Programming Manual Fundamentals; Chapter: "feed rate control" and "spindle motion"

Note

The maximum number of positioning axes that can be programmed in a block is limited to the maximum number of available channel axes.

Definition

Positioning axes are defined using the following parameters:

- Axis type: Positioning axis type 1, type 2 or type 3
- End point coordinates (in absolute dimensions or in incremental dimensions)
- Feedrate for linear axes in [mm/min], for rotary axes in [degrees/min]

Example: Positioning axis type 1

Program code	Comment
`POS[Q1]=200 FA[Q1]=1000`	`; Axis Q1 with feedrate 1000mm/min at Position 200.`

Example: Positioning axis type 2

Program code	Comment
`POSA[Q2]=300 FA[Q2]=1500`	`; Axis Q2 with feedrate 1,500mm/min at Position 300.`

Note

Within a part program, an axis can be a path axis or a positioning axis. Within a movement block, however, each axis must be assigned a unique axis type.

Programming in synchronized action

Axes can be positioned completely asynchronously to the part program from synchronized actions.

Example:

```
Program code                                          Comment
ID=1 WHENEVER $R==1 DO POS[Q4]=10 FA[Q3]=990          ; The axial feedrate is
                                                      specified permanently.
```

References:
Programming Manual, Job Planning; Chapter "Motion synchronized actions":

Block change

The block change can be adjusted for positioning axis types 1 and 2 with:

`FINEA=<axis identifier>` or
`FINEA[<axis identifier>]`

`COARSEA=<Axis identifier>` or
`COARSEA[<axis identifier>]`

`IPOENDA=<axis identifier>` or
`IPOENDA[<axis identifier>]`

In Type 3 positioning axis, the block change within the brake ramp of the single interpolation can be set with:

`IPOBRKA=<axis identifier>` or
`IPOBRKA(<axis identifier>[,<instant in time*>])`

* Instant in time of the block change, referred to the braking ramp as a %

Absolute dimension/incremental dimension

The programming of the end point coordinates takes place in absolute dimension (`G90`) or in incremental dimension (`G91`).

Example	Description
	Programming the end point coordinates
`G90 POS[Q1]=200`	In absolute dimension
`G91 POS[Q1]=AC(200)`	In absolute dimension
`G91 POS[Q1]=200`	In incremental dimension
`G90 POS[Q1]=IC(200)`	In incremental dimension

Reprogram type 2 positioning axes

With type 2 positioning axes (motion across block limits), you need to be able to detect in the part program whether the positioning axis has reached its end position. Only then is it possible to reprogram this positioning axis (otherwise an alarm is issued).

If `POSA` is programmed, then `POSA` again with `IPOBRKA` (block change in the braking ramp), an alarm is not issued. For more information, please refer to NC command `IPOBKA` in Chapter "Settable block change time".

Coordination (WAITP)

The coordination command `WAITP` enables you to designate a position in the NC program where the program is to wait until an axis programmed with `POSA` in a previous NC block has reached its end position.

`WAITP` exists in an internal block.

An explicit reference must be made to any axis for which the program is to wait.

Example:

```
Program code                                          Comment
N10 G01 G90 X200 F1000 POSA[Q1]=200 FA[Q1]=500
N15 X400
N20 WAITP(Q1)                                         ; The program processing
                                                        is stopped automatically
                                                        till Q1 is in position.
N25 X600 POS[Q1]=300                                  ; Q1 is a positioning axis
                                                        of Type 1 (feedrate
                                                        FA[Q1] from block N10).
N30 X800 Q1=500                                       ; Q1 is path axis (path
                                                        feed F1000 from block
                                                        N10).
```

Tool offset

A tool length compensation for positioning axes can be implemented by means of an axial zero offset, allowing, for example, the positioning path of a loader to be altered. An example where the axial zero offset might be used in place of the tool length compensation is where a loader containing tools of various dimensions has to bypass an obstacle.

End of program

The program end (program status selected) is delayed until all axes (path axes + positioning axes) have reached their programmed end points.

11.5.2 Revolutional feed rate in external programming

The two following setting data can be used to specify that the revolutional feed rate of a positioning axis should be derived from another rotary axis/spindle:

SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE(revolutional feed rate for position axes/spindles)

SD42600 JOG_FEED_PER_REV_SOURCE (control of revolutional feed rate in JOG)

The following settings are possible:

Value	Description
0	No revolutional feed rate selected
>0	The revolutional feed rate is derived from the round axis/spindle with the machine axis index specified here.
-1	The revolutional feed rate is derived from the master spindle of the channel in which the axis/spindle is active.
-2	The revolutional feed rate is derived from the rotary axis/spindle with the machine axis index 0.
-3	The revolutional feed rate is derived from the master spindle of the channel in which the axis/spindle is active. No revolutional feed rate is selected if the master spindle is at a standstill.

11.6 Block change

Positioning axes can be programmed in the NC block individually or in combination with path axes.

Path axes and positioning axes are always interpolated separately (path interpolator and axis interpolators) and this causes them to reach their programmed end positions at different times.

There are three types of positioning axis, whose response differs with respect to block change:

- Type 1

  The block change occurs when all path and positioning axes have reached their programmed end point.

- Type 2

  Block change, if all path axes have reached their programmed end points according to G601, G602, G603.

- Type 3

  Block change is settable within the braking ramp of the single axis interpolation if the criteria for the motion end and the block change are fulfilled for the path interpolation.

Positioning axis type 1

Block change at the programmed end point of all path axes and positioning axes

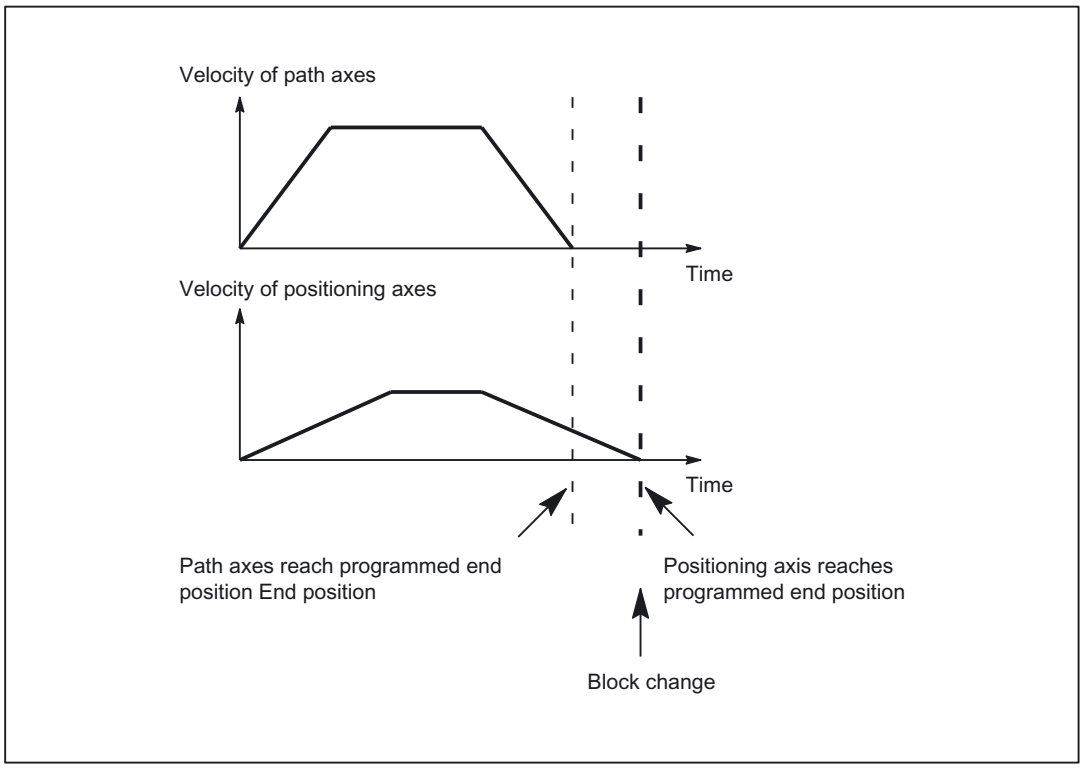


Figure 11-1 Block change with positioning axis type 1, example of sequence

Properties of type 1 positioning axis

With SW 5 and lower, type 1 positioning axes have the following behavior:

- The block change occurs (NC block finished) when all the path and positioning axes have reached the respective end-of-motion criterion.
- Continuous-path mode (G64) is only possible for path axes if the positioning axes reach their end-of-motion criterion ahead of the path axes (this is not the case in the example in the diagram above).
- Programming with

  POS[Name] = end point FA[Name] = feed

  or abbreviated with

  POSA[Name] = end point

  in which case the feed is determined by the setting in MD32060 $MA_POS_AX_VELO.
- The programmed instruction is effective on a non-modal basis. The geometry and synchronous axes are separated with the instructions from the path axis grouping and traversed at an axis-specific velocity.

With SW 5 and higher, type 1 positioning axes have the following additional behavior:

- With continuous-path mode (G601, G602, G603) positioning axes/spindles traverse to the positioning end if end-of-motion criterion FINEA, COARSA or IPOENDA is fulfilled.

Positioning axis type 2

Block change at programmed end point of all path axes

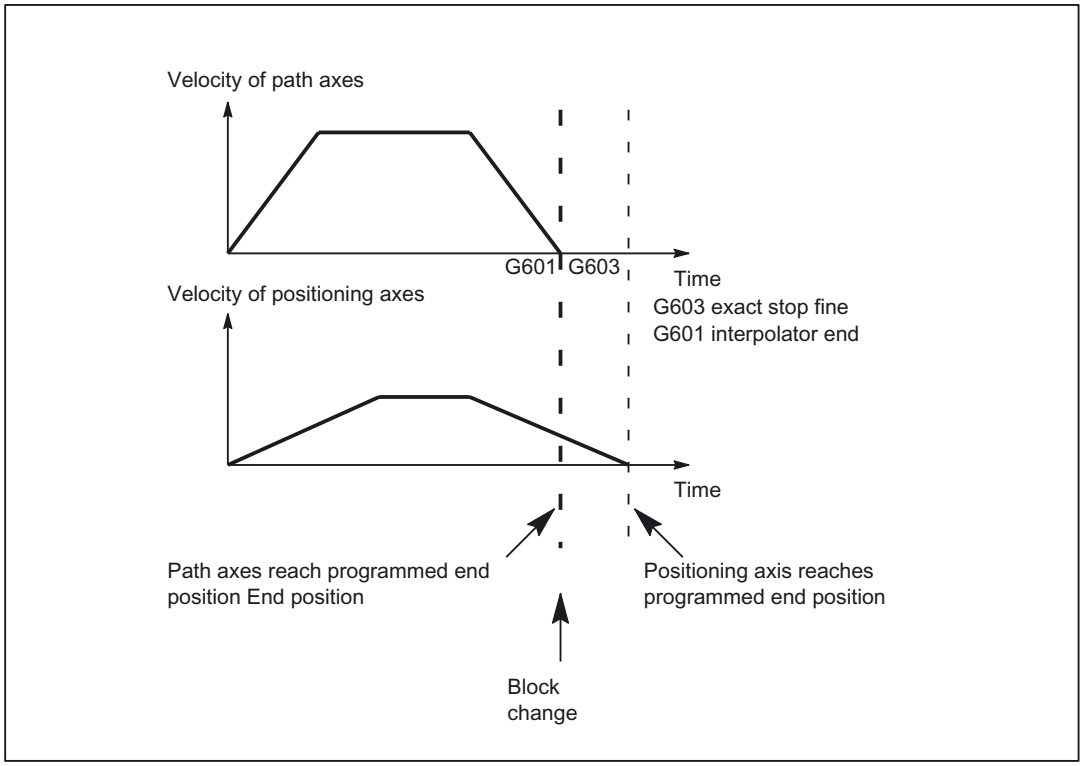


Figure 11-2 Block change with positioning axis type 2, example of sequence

Properties of type 2 positioning axis

With SW 5 and lower, type 2 positioning axes have the following behavior:

- The block change occurs (NC block finished) when the path axes have reached their programmed end positions with regard to G601, G602 and G603.
- The positioning axes can traverse across several block boundaries to their programmed end positions.
- Since there is no time relationship between "NC block finished" and the point at which type 2 positioning axes reach their programmed end positions, the WAITP coordination command is provided for the synchronization of the positioning axes (see Section "Programming").
- If a positioning axis is reprogrammed before it has reached the previous position and before the end-of-motion criterion is fulfilled, the "axis cannot be repositioned" alarm is issued.

- Programming with

  POSA[Name] = end point FA[Name] = feed

  or abbreviated with

  POSA[Name] = end point

  in which case the feed is determined by the setting in MD32060 $MA_POS_AX_VELO.
- The programmed instruction is effective on a non-modal basis. The geometry and synchronous axes are separated with the instructions from the path axis grouping and traversed at an axis-specific velocity.

11.6.1 Settable block change time

Positioning axis type 3

Settable block change time for single axis interpolation

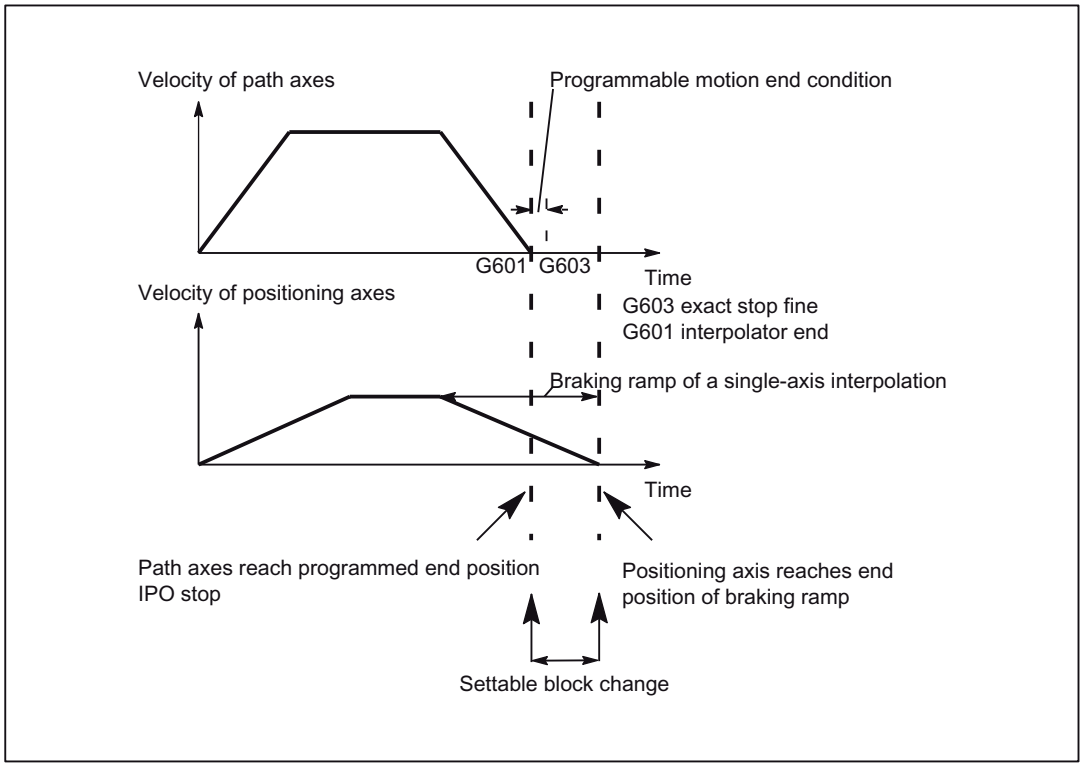


Figure 11-3 Settable block change for type 3 positioning axis, example

Properties of type 3 positioning axis

With type 3 positioning axes the end-of-motion criterion can be programmed with FINEA, COARSEA or IPOENDA. The block change condition can be set within the braking ramp of the single-axis interpolation.

- The block change occurs (NC block finished) when all the path axes/spindles have fulfilled the programmed end-of-motion criteria. The positioning axis/spindle must also have met the programmed block change condition within the braking ramp of the single-axis interpolation. The block change can only be executed if both conditions are fulfilled.
- The position end of the positioning axes can be defined in the programmable end-of-motion criterion by means of one of the following NC commands

  FINEA[axis]: “Exact stop fine” motion end or

  COARSEA[axis]: “Exact stop coarse" motion end or

  IPOENDA[axis]: End of motion when “IPO stop” is reached
- The following syntax applies for the position end of positioning spindles:

  FINEA[Sn]: “Exact stop fine” motion end or

  COARSEA[Sn] "Exact stop coarse" motion end or

  IPOENDA[Sn]: End of motion when “IPO stop” is reached

  Sn: Spindle number, 0 ... max. spindle number or

  Axis: Axis identifiers X, Y, Z.

Command axes

Command axes respond immediately to any reprogramming; therefore, programming should not be changed. Each individual axis has its own axis interpolator and its own feedrate. An axis cannot be moved from the part program and from synchronized actions simultaneously.

The FINEA, COARSEA and IPOENDA end-of-motion criteria are applicable for the transition to a neutral axis. The axis can then be used via the NC program, for example, as a PLC axis.

Reciprocating axes

Reciprocating axes always brake at their reversal position and then move in the opposite direction. Therefore, reciprocating axes do not require an expansion.

Note

The behavior of PLC axes at block changes is described in Section "Control by the PLC".

For further information about block changes with programmed end-of-motion criteria `FINEA`, `COARESA` and `IPOENDA`, please refer to:

References:

/FB1/ Function Manual, Basic Functions; Spindles (S1), Section "Spindle modes"

/FB1/ Function Manual, Basic Functions; Feedrates (V1), Section "Programmable dynamic response of single axes"

IPOBRKA

When the NC command `IPOBRKA` is issued, the end-of-motion criterion is activated in the braking ramp either

- in the part program: Block change in the part program or
- in synchronized action: Technology cycles

With SW 6.2 and higher, type 2 positioning axis behavior differs from that in previous software versions as follows:

If `POSA` is programmed, then `POSA` again with `IPOBRKA` (block change in the braking ramp), an alarm is not issued. The new `POSA` motion waits for the preceding `POSA` motion to have reached the block change point in the braking ramp.

Condition for block change

If the end-of-motion criteria for all axes/spindles to be operated in the block are fulfilled in addition to the block change condition, then block change takes place. This applies both to part program blocks and for technology cycle blocks.

Block change condition braking ramp in SD43600

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE

The above setting data "Block change condition braking ramp" can be set to define the moment of block change for single-axis interpolation to between the activation instant of the braking ramp with 100% to IPO stop with 0%.

With a setting of 100%, the block change condition of the positioning axis is already fulfilled when the braking ramp is activated.

With a setting of 0%, the block change condition is identical to `IPOENDA`.

Advantages of the percent setting

Setting SD43600 in percent offers the following advantages:

- The block change condition is not dependent on a position and is therefore dependent on the override set.
- Maximum override will result in the greatest smoothing deviation.
- A smaller override will reduce the deviation.

The braking ramp block change condition can be queried like the previous axial end-of-motion criteria with$AA_MOTEND[axis] = 4

Braking ramp with tolerance window

Additional block change condition in braking ramp

An additional block change condition tolerance window can be selected as well as the existing block change condition in the braking ramp. Release will only occur when the axis

- as before has reached the specified % value of its braking ramp **and**
- its current actual or set position is no further than an adjustable tolerance from the end of the axis in the block.

The latest the axis is available is when the interpolator reaches the end position. The tolerance window is modal.

The following applies to the braking ramp **with tolerance window** block change condition:

The setpoint position is signaled with $AA_MOTEND[axis] = 5

The actual position is signaled with $AA_MOTEND[axis] = 6

Supplementary conditions

Block change and alteration of axis status

A premature block change is not possible:

- During oscillation with partial infeed

  The block-specific oscillation motion must remain active until the axis with partial infeed has reached its end position.

- During handwheel operation

  The last set end-of-motion criterion applies.

The axis for which a block change occurred within the braking ramp can only be programmed in the following block in the same axis state.

When the axis state changes, e.g. to POS from SPOS, the last programmed end-of-motion criterion FINEA, COARSEA, IPOENDA applies.

This also applies:

- if a positioning axis is changed to a path axis or
- if the program is waiting specifically for the end of the positioning movement

  (WAITP, M30, technology cycle end, preprocessing stop).

- if a velocity override is activated or deactivated.

Activation and deactivation

End-of-motion criterion IPOBRKA and precise time of activation.

For part program execution:

- The braking ramp end-of-motion criterion can be activated via the NC command IPOBRKA.
- The precise time of activation is defined in setting data

  SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE
- `IPOBRKA` is deactivated when an axis end-of-motion criterion `FINEA`, `COARSEA` or `IPOENDA` is next programmed for the axis.
- The NC command ADISPOSA can be used to program the size of the tolerance window for the braking ramp end-of-motion criterion and this is defined by means of setting data SD43610 $SA_ADISPOSA_VALUE[axis]:

For technology cycles:

- The braking ramp end-of-motion criterion can be activated via the synchronized action `IPOBRKA`.
- `IPOBRKA` is deactivated when the axis end-of-motion criterion `FINEA`, `COARSEA` or `IPOENDA` is next programmed for the axis in a synchronized action.
- The size of the tolerance window for the braking ramp end-of-motion criterion can be programmed via the synchronized action ADISPOSA.

Note

For further information about programming positioning axes, see:

References:

/PG/ Programming Manual, Fundamentals, Section "Feedrate control and spindle motion"

/PGA/, Programming Manual, Job Planning, Section "Special motion commands"

Examples

For block change condition "Braking ramp" in the **part program**:

```
                        ; Default effective
N10 POS[X]=100          ; Block change occurs when the X axis reaches
                        ; position 100 and exact stop fine.
N20 IPOBRKA(X,100)      ; Braking ramp block change condition
N30 POS[X]=200          ; Block change occurs as soon as the X axis
                        ; begins to brake
N40 POS[X]=250          ; X axis does not brake at position 200,
                        ; but continues moving
                        ; to position 250, as soon as the X axis
                        ; begins to brake, the block changes.
```

```
N50 POS[X]=0            ; the X axis brakes and traverses back
                        ; to position 0, the block change
                        ; takes place at position 0 exact stop fine
N60 X10 F100
N70 M30
```

At direction reversal (N50), the axis first brakes to reach the target position, before it can be traversed in the opposite direction.

Alternatively, IPOBRKA(X) can also be written in block N20 if the value 100 has already been entered in

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE[AX1].

For block change condition "Braking ramp" in **synchronized action**:

In the technology cycle:

```
FINEA                   ; End of motion criterion fine exact stop
POS[X]=100              ; Technology cycle block change takes place
                        ; when the X axis has reached position 100
                        ; and exact stop fine.
IPOBRKA(X,100)          ; Activate block change criterion braking ramp
POS[X]=200              ; Technology cycle block change takes place
                        ; as soon as X axis starts to brake.
POS[X]=250              ; X axis does not brake at position 200,
                        ; but continues traversing to position 250
N40 POS[X]=250          ; As soon as the X axis starts to brake,
                        ; the block change occurs in the technology cycle.
POS[X]=0                ; The X axis brakes and returns to position 0
M17                     ; the block change takes place at position 0 exact stop fine.
```

With tolerance window

For block change condition "Braking ramp" in the **part program**:

```
                          ; Default effective
N10 POS[X]=100            ; Block change occurs when the X axis reaches
                          ; position 100 and exact stop fine.
N20 IPOBRKA(X,100)        ; Braking ramp block change condition
N21 ADISPOSA(X,1,0.5)     , Define tolerance window setpoint position for X axis
                          ; as 0.5
N30 POS[X]=200            ; Block change occurs as soon as the X axis
                          ; begins to brake and the
                          ; setpoint position is ≥ 199.5.
N40 POS[X]=250            ; X axis does not brake at position 200,
                          ; but continues traversing to position 250,
                          ; as soon as the X axis starts to brake,
                          ; and the position is ≥ 249.5,
                          ; the block change takes place.
```

```
N50 POS[X]=0              ; the X axis brakes and traverses back
                          ; to position 0
                          ; the block change occurs at
                          ; position 0 exact stop fine
N60 X10 F100
N70 M30
```

At direction reversal (N50), the axis first brakes to reach the target position, before it can be traversed in the opposite direction.

For block change condition "Braking ramp" in **synchronized action**:

In the technology cycle:

```
FINEA                     ; End of motion criterion fine exact stop
POS[X]=100                ; Technology cycle block change takes place
                          ; when the X axis has reached position 100
                          ; and exact stop fine.
ADISPOSA(X,2,0.3)         ; Define tolerance window actual position for
                          ; X axis as 0.3
IPOBRKA(X,100)            ; Activate block change criterion braking ramp
POS[X]=200                ; Technology cycle block change takes place
                          ; as soon as the X axis starts to brake and
                          ; the X axis actual position is ≥ 199.7.
POS[X]=250                ; X axis does not brake at position 200,
                          ; but continues moving
N40 POS[X]=250            ; to position 250, as soon as the X axis
                          ; begins to brake and reaches position
                          ; ≥ 249.7, the block changes in the
                          ; technology cycle.
POS[X]=0                  ; the X axis brakes and traverses back
                          ; to position 0
M17                       ; the block change occurs at position 0
                          ; and exact stop fine.
```

11.6.2 End of motion criterion with block search

Last block serves as container

The last end-of-motion criterion programmed for an axis is collected and output in an action block. The last block with a programmed motion end condition that was processed in the search run serves as a container for setting all axes.

Example

For two action blocks with end-of-motion criteria for three axes:

```
N01 G01 POS[X]=20 POS[Y]=30
 IPOENDA[X]                      ; last programmed
                                 ; end-of-motion criterion IPOENDA
N02 IPOBRKA(Y, 50)               ; second action block for
                                 ; the Y axis IPOENDA
N03 POS[Z]=55 FINEA[Z]           ; second action block for the Z axis FINEA
N04 $A_OUT[1]=1                  ; first action block for output
                                 ; as digital output
N05 POS[X]=100                   ;
N06 IPOBRKA(X, 100)              ; second action block for
                                 ; the X axis IPOBRKA container
...                              ; for all programmed
                                 ; end-of-motion criteria
TARGET:                          ; Block search target
```

The first action block contains the digital output:

$A_OUT[1] = 1.

The second action block contains the settings for the end-of-motion criteria:

for the X axis IPOBRKA / $SA_IPOBRAKE_BLOCK_EXCHANGE[AX1]=100

for the Y axis IPOBRKA / $SA_IPOBRAKE_BLOCK_EXCHANGE[AX2]=50

for the Z axis FINEA. The motion end condition IPOENDA last programmed is also stored for the X axis.

11.7 Control by the PLC

PLC axes

PLC axes are traversed from the PLC and can move asynchronously to all other axes. The travel motions are executed separate from the path and synchronized actions.

Reference:
Function Manual Basic Functions; "Basic PLC Program for SINUMERIK 840D sl" (P3) or "PLC for SINUMERIK 828D" (P4)

Concurrent positioning axes

Using function block FC18, for SINUMERIK 840D sl concurrent positioning axes can be started from the PLC.

Channel-specific signals

All channel-specific signals act to the same extent on path and positioning axes.

Only the following signals are an exception:

- IS DB21, ... DBB4 ("Feedrate override")
- IS DB21, ... DBX6.2 ("Delete distance to go")

Axis-specific signals

Positioning axes have the following additional signals:

- IS DB31, ... DBX76.5 ("Positioning axis")
- Feedrate for positioning axes/spindles (FA)
- IS DB31, ... DBB0 ("Feedrate override") axis-specific
- IS DB31, ... DBX2.2 ("Distance to go/Spindle reset") Axis-specific deletion of distance to go

Single-axis functions of PLC-controlled axes

The behavior of individual PLC axes can be manipulated in the following way with machine data:
MD30460 $MA_BASE_FUNCTION_MASK:

- Bit 4 = 1

  The axis is exclusively controlled by the PLC.

- Bit 5 = 1

  The axis is a permanently assigned PLC axis.

- Bit 6 = 1

  The channel-specific NC/PLC interface signal:
  DB21, ... DBX6.0 ("feed disable")
  is not active for the axis if this is a PLC-controlled axis.

- Bit 7= 1

  The channel-specific NC/PLC interface signal:
  DB21, ... DBX36.3 ("all axes stationary")
  is set **independently** of an axis if this is a PLC-controlled axis.

For a PLC-controlled axis:

- The channel-specific NC/PLC interface signal DB21, ... DBX6.0 ("feed disable"), is active if bit 6 = 0 in machine data MD30460 $MA_BASE_FUNCTION_MASK.
- The channel-specific NC/PLC interface signal DB21, ... DBX36.3 ("all axes stationary") is only set if bit 7 = 0 in machine data MD30460 $MA_BASE_FUNCTION_MASK.

If an attempt is made to assign an **exclusively PLC-controlled axis** to the NC program or to request the axis for the NC program, then this is rejected with Alarm 26075. Alarm 26076 is generated in the same way for a PLC axis with fixed assignment.

A **PLC axis with fixed assignment** is a "neutral axis" on power up. For a travel request via the NC/PLC interface, a concurrent positioning axis automatically changes to a PLC axis without being interchanged beforehand.

Axis replacement via PLC

The type of an axis for axis replacement is transferred to the PLC with axial interface byte NCK→PLC NST DB31, ... DBB68:

- IS DB31, ... DBX68.0-68.3 ("NC axis/spindle in channel") channels 1 to 10
- IS DB31, ... DBX68.4 ("new type requested by PLC")
- IS DB31, ... DBX68.5 ("axis can be replaced")
- IS DB31, ... DBX68.6 ("neutral axis/spindle")
- IS DB31, ... DBX68.7 ("PLC axis/spindle")

If IS DB31, ... DBX68.5 ("axis can be replaced") == 1, an axis replacement request can be issued from the PLC.

References:
Function Manual, Extended Functions; Mode Groups, Channels, Axis Replacement (K5)

11.7.1 Starting concurrent positioning axes from the PLC

Activation from PLC

When concurrent positioning axes are activated from the PLC, FC18 is called and supplied with the following parameter data:

- Axis name or axis number
- End position
- Feedrate (with feedrate setting = 0, the feedrate is determined by the setting in machine data MD32060 $MA_POS_AX_VELO):
- Absolute coordinate (G90),
  incremental coordinate (G91),
  absolute coordinate along the shortest path for rotary axes (rotary axis name = DC(value))

The following functions are defined:

- Linear interpolation (G01)
- Feedrate in mm/min or degrees/min (G94)
- Exact stop (G09)
- Settable zero offsets currently selected are valid

Since each axis is assigned to exactly one channel, the control can select the correct channel from the axis name/axis number and start the concurrent positioning axis on this channel.

11.7.2 PLC-controlled axes

PLC actions

The table below compares the following PLC actions with the corresponding NCK reactions for a machine axis 1:

- Start machine axis as PLC axis via FC 18
- Trigger NC Start or NC Stop
- Trigger axial `STOP`, `RESUME` or `RESET`
- Trigger `NC-RESET`
- Cancel or set servo enable for the machine axis
- Relinquish control of machine axis to NC

Examples of NCK reactions

PLC actions are shown as NCK reactions in the table below.

PLC actions	NCK reaction
Start machine axis 1, residing in 1st channel, as PLC axis via FC 18	
Trigger IS DB21, ... DBX7.4 (”NC Stop axes plus spindle”)	Machine axis 1 is stopped.
Trigger IS DB21, ... DBX7.1 (”NC Start”)	Machine axis 1 continues traversal.
PLC wants to control machine axis 1, IS DB31, ... DBX28.7 ==1 (”PLC controls axis”)	Control of machine axis 1 is given to the PLC. IS DB31, ... DBX63.1==1 ("PLC controls axis")
Trigger IS DB21, ... DBX7.4 (”NC Stop axes plus spindle”)	Machine axis 1 is not stopped.
Trigger axial axis stop IS DB31, ... DBX28.6 (”AxStop, Stop”)	Machine axis 1 is stopped IS DB31, ... DBX63.2==1 (”axis stop active")
Trigger axial RESUME IS DB31, ... DBX28.2 (”AXRESUME”)	Machine axis 1 continues traversing IS DB31, ... DBX63.2==0 (”axis stop active")
Trigger NC Reset Trigger IS DB21, ... DBX7.7 (”Reset”)	No effect on machine axis 1
Trigger axial RESET IS DB31, ... DBX28.1 (”AXRESET”)	Machine axis 1 is stopped and the traversing movement is aborted. IS DB31, ... DBX63.2==1 (”axis stop active”) is reset to 0, its axial machine data are read in, IS DB31, ... DBX63.0 (”AXRESET DONE”) is set to 1 and IS DB31, ... DBX63.2 (”axis stop active”) is reset to 0.
Start machine axis 1 as PLC axis via FC 18	IS DB31, ... DBX63.0==0 (”AXRESET DONE”)
Cancel servo enable for machine axis 1 IS DB31, ... DBX2.1==0 (”servo enable”)	Axial alarm 21612 ”Axis %1 measuring system change" is signaled.
Trigger axial RESUME IS DB31, ... DBX28.2 (”AXRESUME”)	Axial alarm 21612 ("Axis %1 measuring system change") is canceled and motion command IS DB21, ... DBX40.7 ("Motion command plus") is output. But machine axis 1 does not start to traverse due to a missing servo enable.
Set servo enable for machine axis 1 IS DB31, ... DBX2.1==1 (”servo enable”)	Machine axis 1 traverses to the programmed end point.
Trigger axial RESET IS DB31, ... DBX28.1 (”AXRESET”)	Machine axis 1 is stopped and traversal aborted, its axial machine data are read in and the end of axial RESET IS DB31, ... DBX63.0==0 (”AXRESET DONE”) is signaled.
PLC releases control of machine axis 1 to the NCK. IS DB31, ... DBX28.7==0 ("PLC controls axis")	NCK accepts control of the machine axis IS DB31, ... DBX63.1==0 (”PLC controls axis”) and resets the end signal of axial RESET IS DB31, ... DBX63.0 (”AXRESET DONE”) from 1 to 0.

11.7.3 Control response of PLC-controlled axes

Response to channel reset, NEWCONFIG, block search and MD30460

Control response to	PLC-controlled axis
Mode change and NC program control	work independently of axis.
Channel RESET	No axial machine data are effective and a traversing movement is not aborted.
NEWCONFIG	No axial machine data are effective.
Block search Type 5 SERUPRO	are processed during SERUPRO to simulate the normal procedure, e.g. PLC takes over or relinquishes control of this axis which is also traversing via the PLC.
All block search variants of types 1, 2, 4 and 5	The PLC takes over control of the axis **before** the approach block and is responsible for positioning this axis.
NC-controlled retraction activated with $AC_ESR_TRIGGER.	has no effect and acts only on the specified PLC-controlled axis.
machine data: MD30460 $MA_BASE_FUNKTION_MASK	**which is not controlled exclusively by the PLC**
Bit 4 = 0	**cannot** be changed directly with axis replacement command GET (axis) or AXTOCHAN(axis, channel) to an axis controlled by the NC program, see *** Note on axis replacement**.
Bit 4 = 1	cannot be requested for the NC program. GET or AXTOCHAN from the NC program or a synchronized action, or programming the axis in the NC program, are rejected with alarm 26075.
MD30460 $MA_BASE_FUNKTION_MASK	**For the PLC-controlled axis**
Bit 6 = 1	channel-specific IS DB 21, ... DBX6.0 ("feed disable") is not effective. This axis is not stopped when feed disable is activated, but continues to move.
Bit 7 = 1	the axis is not taken into account when IS DB 21, ... DBX36.3 ("all axes stationary") is generated. This signal is output with 1 even if all other axes in the channel are stationary and only the PLC-controlled axis is active.

*** Note on axis replacement**

This replacement of a "neutral axis" by an "NC program axis" does not take place until the PLC has really relinquished control of the axis in accordance with use case "Relinquish control of axis". Waiting for this axis interchange is displayed on the HMI operator panel front.

11.8 Response with special functions

11.8.1 Dry run (DRY RUN)

The dry run feedrate is also effective for positioning axes unless the programmed feedrate is larger than the dry run feedrate.

Activation of the dry run feed entered in SD42100 $SA_DRY_RUN_FEED can be controlled with SD42101 $SA_DRY_RUN_FEED_MODE. See

References:

/FB1/ Function Manual, Basic Functions; Feedrates (V1)

11.8.2 Single block

Positioning axis type 1

Single-block mode is effective with positioning axes of type 1.

Positioning axis type 2

Positioning axes of type 2 also continue across block limits in single block mode.

Positioning axis type 3

Positioning axes of type 3 also continue across block limits in single block mode.

11.9 Examples

11.9.1 Motion behavior and interpolation functions

In the following example, the two positioning axes Q1 and Q2 represent two separate units of movement. There is no interpolation relationship between the two axes. In the example, the positioning axes are programmed as type 1 (e.g. in N20) and type 2 (e.g. in N40).

Program example

```
N10 G90 G01 G40 T0 D0 M3 S1000
N20 X100 F1000 POS[Q1]=200 POS[Q2]=50 FA[Q1]=500
FA[Q2]=2000
N30 POS[Q2]=80
N40 X200 POSA[Q1] = 300 POSA[Q2]=200] FA[Q1]=1500
N45 WAITP[Q2]
N50 X300 POSA[Q2]=300
N55 WAITP[Q1]
N60 POS[Q1]=350
N70 X400
N75 WAITP[Q1, Q2]
N80 G91 X100 POS[Q1]=150 POS[Q2]=80
N90 M30
```

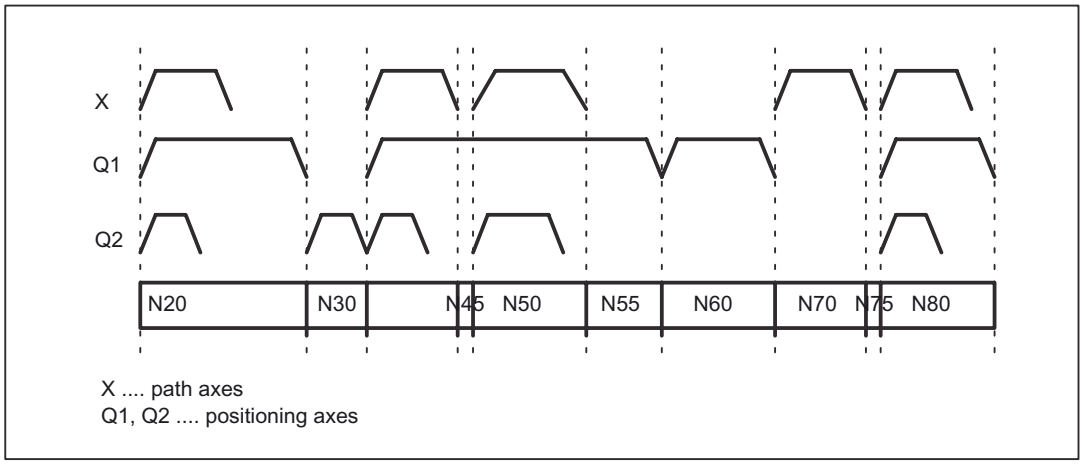


Figure 11-4 Timing of path axes and positioning axes

11.9.1.1 Traversing path axes without interpolation with G0

Example in G0 for positioning axes

Path axes traverse as positioning axes with no interpolation in rapid traverse mode (G0):

```
                                ; activation of nonlinear
                                ; interpolation
                                ; MD20730 $MC_GO_LINEAR_MODE == FALSE
                                ; is set
G0 X0 Y10                       ; axis traverses without interpolation
G0 G43 X20 Y20                  ; axis traverses in path mode (with interpolation)

G0 G64 X30 Y30                  ; axis traverses in path mode (with interpolation)

G0 G95 X100 Z100 m3 s100        ; axis traverses without interpolation
                                ; no revolutional feedrate active
```

11.10 Data lists

11.10.1 Machine data

11.10.1.1 Channel-specific machine data

Number	Identifier: $MC_	Description
20730	G0_LINEAR_MODE	Interpolation behavior with G0
20732	EXTERN_G0_LINEAR_MODE	Interpolation behavior with G00
22240	AUXFU_F_SYNC_TYPE	Output timing of F functions

11.10.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30450	IS_CONCURRENT_POS_AX	Concurrent positioning axis
30460	BASE_FUNCTION_MASK	Axis functions
32060	POS_AX_VELO	Feedrate for positioning axis
37510	AX_ESR_DELAY_TIME1	Delay time for ESR single axis
37511	AX_ESR_DELAY_TIME2	ESR time for interpolatory braking of single axis

11.10.2 Setting data

11.10.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43600	IPOBRAKE_BLOCK_EXCHANGE	Braking ramp block change condition
43610	ADISPOSA_VALUE	Braking ramp tolerance window

11.10.3 Signals

11.10.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feed disable	DB21, ...DBX6.0	DB3200.DBX6.0
NC Start	DB21, ...DBX7.1	DB3200.DBX7.1

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NC stop axes plus spindle	DB21, ...DBX7.4	DB3200.DBX7.4
Reset	DB21, ...DBX7.7	-

11.10.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
All axes stationary	DB21, ...DBX36.3	DB3300.DBX4.3
Travel command minus	DB21, ...DBX40.6	DB3300.DBX1000.6
Travel command plus	DB21, ...DBX40.7	DB3300.DBX1000.7

11.10.3.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Feedrate override, axis-specific	DB31, ...DBB0	DB380x.DBB0
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Delete distance-to-go spindle reset for specific axes	DB31, ...DBX2.2	DB380x.DBX2.2
AXRESET	DB31, ...DBX28.1	-
AXRESUME	DB31, ...DBX28.2	-
AxStop, stop	DB31, ...DBX28.6	-
PLC-controlled axis	DB31, ...DBX28.7	-

11.10.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Exact stop coarse	DB31, ...DBX60.6	DB390x.DBX0.6
Exact stop fine	DB31, ...DBX60.7	DB390x.DBX0.7
Axial alarm	DB31, ...DBX61.1	DB390x.DBX1.1
Axis ready (AX_IS_READY)	DB31, ...DBX61.2	DB390x.DBX1.2
Axis container rotation active	DB31, ...DBX62.7	-
AXRESET DONE	DB31, ...DBX63.0	-
PLC-controlled axis	DB31, ...DBX63.1	-
Axis stop active	DB31, ...DBX63.2	DB390x.DBX3.2
Travel command minus	DB31, ...DBX64.6	DB390x.DBX4.6
Travel command plus	DB31, ...DBX64.7	DB390x.DBX4.7
Positioning axis	DB31, ...DBX76.5	DB390x.DBX1002.5
F function (feedrate) for positioning axis	DB31, ...DBB78-81	-
Emergency retraction active	DB31, ...DBX98.7	DB390x.DBX5002.7

P5: Oscillation - only 840D sl

12

12.1 Brief description

Definition

When the "Oscillation" function is selected, an oscillation axis oscillates backwards and forwards at the programmed feedrate or a derived feedrate (revolutional feedrate) between two reversal points. Several oscillation axes can be active at the same time.

Oscillation variants

Oscillation functions can be classified according to the axis response at reversal points and with respect to infeed:

- Asynchronous oscillation across block boundaries

  During reciprocating movement, other axes can interpolate at will. The oscillation axis can act as the input axis for dynamic transformation or as the master axis for gantry or coupled-motion axes. Oscillation is not automatically linked to the AUTOMATIC mode.

- Oscillation with continuous infeed.

  Simultaneous infeed in multiple axes is possible. However, there is no interpolative connection between the infeed and oscillation movements.

- Oscillation with infeed in both reversal points or only in the left-hand or right-hand reversal point. The infeed can be initiated at a programmable distance from the reversal point.
- Sparking-out strokes after oscillation.
- Beginning and end of oscillation at defined positions.

Response at reversal points

The change in direction is initiated:

- without the exact stop limit being reached (exact stop fine or coarse)
- after reaching the programmed position or
- after the programmed position is reached and expiry of a dwell.
- by an external signal (from the PLC).

Control methods

Oscillation movements can be controlled by various methods:

- The oscillation movement and/or infeed can be interrupted by delete distance-to-go.
- The reversal points can be altered via NC program, PLC, HMI, handwheel or directional keys.
- The feedrate velocity of the oscillation axis can be altered through a value input in the NC program, PLC, HMI or via an override. The feedrate can be programmed to be dependent on a master spindle, rotary axis or spindle (revolutional feedrate).

  References:

  /FB1/ Function Manual Basic Functions; Feedrates (V1)
- The oscillation movement can be controlled entirely by the PLC.

Methods of oscillation control

There are two modes of oscillation:

1. Asynchronous oscillation:

   Is active across block boundaries and can also be started from the PLC/HMI.
2. Oscillation by synchronized movement actions:

   In this case the asynchronous oscillation and an infeed movement are coupled via synchronized actions. In this way, it is possible to program oscillation with infeed at the reversal points which is active on a non-modal basis.

12.2 Asynchronous oscillation

Characteristics

The characteristics of asynchronous oscillation are as follows:

- The oscillation axis oscillates backwards and forwards between reversal points at the specified feedrate until the oscillation movement is deactivated or until there is an appropriate response to a supplementary condition. If the oscillation axis is not positioned at reversal point 1 when the movement is started, then it traverses to this point first.
- Linear interpolation G01 is active for the oscillation axis regardless of the G code currently valid in the program. Alternately, revolutional feedrate G95 can be activated.
- Asynchronous oscillation is active on an axis-specific basis beyond block limits.
- Several oscillation axes (i.e. maximum number of positioning axes) can be active at the same time.
- During the oscillation movement, axes other than the oscillation axis can be freely interpolated. A continuous infeed can be achieved via a path movement or with a positioning axis. In this case, however, there is no interpolative connection between the oscillation and infeed movements.
- If the PLC does not have control over the axis, then the axis is treated like a normal positioning axis during asynchronous oscillation. In the case of PLC control, the PLC program must ensure via the appropriate stop bits of the VDI interface that the axis reacts in the desired way to VDI signals. These signals include program end, operating mode changeover and single block.
- The oscillation axis can act as the input axis for the transformations (e.g. inclined axis).

  References:

  /FB2/ Function Manual, Extended Functions; Transmit/Surface area transformation (M1)
- The oscillation axis can act as the master axis for gantry and coupled motion axes.

  References:

  /FB3/ Function Manual Special Functions; Gantry Axes" (G1)
- It is possible to traverse the axis with jerk limitation (SOFT) and/or with kneeshaped acceleration characteristic (as for positioning axes).
- In addition to this, the oscillation movement can be activated in synchronism with the block via the parts program.
- The oscillation movement can likewise be started, influenced and stopped from the PLC/HMI.
- Interpolatory oscillation is not possible (e.g. oblique oscillation).

12.2.1 Influences on asynchronous oscillation

Setting data

The setting data required for oscillation can be set with special language commands in the NCK parts program, via the HMI and/or the PLC.

Feedrate

The feed velocity for the oscillation axis is selected or programmed as follows:

- The velocity defined for the axis as a positioning axis is used as the feed velocity. This value can be programmed via FA[axis] and has a modal action. If no velocity is programmed, then the value stored in machine data POS_AX_VELO is used (see positioning axes).
- When an oscillation movement is in progress, the feed velocity of the oscillation axis can be altered via setting data. It can be specified via the parts program and setting data whether the changed velocity must take effect immediately or whether it should be activated at the next reversal point.
- The feed velocity can be influenced via the override (axial VDI signal and programmable).
- If Dry Run is active, the dry run velocity setting is applied if it is higher than the currently programmed velocity.

  Activation of the dry run feed entered in SD42100 $SC_DRY_RUN_FEED can be controlled with SD42101 $SC_DRY_RUN_FEED_MODE. See

  References:

  /FB1/ Function Manual Basic Functions; Feedrates (V1)
- Velocity overlay/path overlay can be influenced by the handwheel.

  See also the "Stop time effect" table.

  References:

  /FB2/ Function Manual, Extended Functions; Manual and Handwheel Travel (H1)
- The oscillation axis can be moved with revolutional feedrate.

Revolutionary feedrate

The reversal feed can also be used for oscillation axes.

Reversal points

The positions of the reversal points can be entered via setting data before an oscillation movement is started or while one is in progress.

- The reversal point positions can be entered by means of manual travel (handwheel, JOG keys) before or in the course of an oscillation movement, regardless of whether the oscillation movement has been interrupted or not.

The following applies to alteration of a reversal point position: When an oscillation movement is already in progress, the altered position of a reversal point does not become effective until this point is approached again. If the axis is already approaching the position, the correction will take effect in the next oscillation stroke.

Note

If a reversal point must be altered at the same time as VDI interface signal *"Activate DRF"* is set, the handwheel signals are applied both to the DRF offset and to the offset of the reversal point, i.e. the reversal point is shifted absolutely by an amount corresponding to twice the distance.

Stop times

A stop time can be programmed via setting data for every reversal point.

The setting can be changed in the following blocks of the NC program. It is then effectively block synchronous from the next reversal point.

Stop time can be changed asynchronously via setting data. It is then effective from the instant that the appropriate reversal point is next traversed.

The following table explains the motional behavior in the exact stop range or at the reversal point, depending on the stop time input.

Table 12- 1 Effect of stop time

Stop time setting	Response
-2	Interpolation continues without wait for exact stop
-1	Wait for coarse exact stop at reversal point
0	Wait for fine exact stop at reversal point
>0	Wait for fine exact stop at reversal point, followed by wait for stop time.

Deactivate oscillation

One of the following options can be set for termination of the oscillation movement when oscillation mode is deactivated:

- Termination of oscillation movement at the next reversal point
- Termination of oscillation movement at reversal point 1
- Termination of oscillation movement at reversal point 2

Following this termination process, sparking-out strokes are processed and an end position approached if programmed.

On switchover from asynchronous oscillation to spark-out and during spark-out, the response at the reversal point regarding exact stop corresponds to the response determined by the stop time programmed for the appropriate reversal point. A sparking-out stroke is the movement towards the other reversal point and back. See table:

Note

Oscillation with motion-synchronous actions and stop times "OST1/OST2"

Once the set stop times have expired, the internal block change is executed during oscillation (indicated by the new distances to go of the axes). The deactivation function is checked when the block changes. The deactivation function is defined according to the control setting for the motional sequence "OSCTRL".

This dynamic response can be influenced by the feed override.

An oscillation stroke may then be executed before the sparking-out strokes are started or the end position approached.

Although it appears as if the deactivation response has changed, this is not in fact the case.

Table 12- 2 Operational sequence for deactivation of oscillation

Function	Inputs	Explanation
Deactivation at defined reversal point	Number of sparking-out strokes equals 0, no end position active	The oscillation movement is stopped at the appropriate reversal point
Deactivation with specific number of sparking-out strokes	Number of sparking-out strokes is not equal to 0, no end position is active	After the appropriate reversal point is reached, the number of sparking-out strokes specified in the command are processed.
Deactivation with sparking-out strokes and defined end position (optional)	Number of sparking-out strokes is not equal 0 end position active	After appropriate reversal point is reached, the number of sparking-out strokes specified in command are processed, followed by approach to specified end position.
Deactivation without sparking-out strokes, but with defined end position (optional)	Number of sparking-out strokes is equal 0 end position active	After appropriate reversal point is reached, axis is traversed to specified end position.

NC language

The NC programming language allows asynchronous oscillation to be controlled from the parts program. The following functions allow asynchronous oscillation to be activated and controlled as a function of NC program execution.

Note

If the setting data are directly written in the parts program, then the data change takes effect prematurely with respect to processing of the parts program (at the preprocessing time). It is possible to re-synchronize the parts program and the oscillation function commands by means of a preprocessing stop (STOPRE).

References:

/PGA/ Programming Manual Job Planning

1) Activate, deactivate oscillation:

- OS[oscillation axis] = 1; Activate oscillation for oscillation axis
- OS[oscillation axis] = 0; Deactivate oscillation for oscillation axis

Note

Every axis may be used as an oscillation axis.

2) End of oscillation:

- WAITP(oscillation axis)

  Positioning axis command – stops block until oscillation axis is at fine stop and synchronizes preprocessing and main run. The oscillation axis is entered as positioning axis again and can then be used normally.

  If an axis is to be used for oscillation, it must be released with a WAITP(axis) command beforehand.

  This also applies if oscillation is initiated from the PLC/HMI. In this case, the WAITP(axis) call is also needed if the axis was programmed beforehand via the NC program. With SW version 3.2 and higher it is possible to select via machine data $MA_AUTO_GET_TYPE, whether WAITP() shall be performed with programming or automatically.

Note

WAITP effectively implements a time delay until the oscillation movement has been executed. Termination of the movement can be initiated, for example, through a programmed deactivation command in the NC program or via the PLC or HMI by means of deletion of distance-to-go.

3) Setting reversal points:

- OSP1[oscillation axis] = position of reversal point 1
- OSP2[oscillation axis] = position of reversal point 2

A position is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed.

If incremental traversal is active, then the position is calculated incrementally to the last appropriate reversal point programmed in the NC program.

4) Stopping times at reversal points:

- OST1[oscillation axis] = stop time at reversal point 1 in [s]
- OST2[oscillation axis] = stop time at reversal point 2 in [s]

A stop time is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed.

The unit for the stop time is identical to the unit selected for the stop time programmed with G04.

5) Setting feedrate:

- FA[axis] = FValue

  Positioning axis infeed.

The feedrate is transferred to the appropriate setting data in synchronism with the block in the main run. If the oscillation axis is moved with revolutional feedrate, the corresponding dependencies must be indicated as described in the Function Manual V1.

6) Setting control settings for sequence of movements:

- OSCTRL[oscillating axis] = (set options, reset options)

  The set options are defined as follows (the reset options deselect the settings):

Table 12- 3 Set/reset options

Option value	Meaning
0	Stop at next reversal point on deactivation of the oscillation movement (default). Can only be achieved by resetting option values 1 and 2.
1	Stop at reversal point 1 on deactivation of the oscillation movement
2	Stop at reversal point 2 on deactivation of the oscillation movement
3	On deactivation of oscillation movement, do not approach reversal point unless sparking-out strokes are programmed
4	Approach an end position after spark-out process
8	If the oscillation movement is aborted by delete distance-to-go, the sparking-out strokes must then be executed and the end position approached (if programmed)
16	If the oscillation movement is terminated by deletion of distance-to-go, the programmed reversal point must be approached on deactivation of the oscillation movement.
32	Altered feedrate will only take effect from the next reversal point.
64	If feedrate setting is 0, path overlay is active, or otherwise velocity overlay
128	For rotary axis DC (shortest path)
256	Sparking-out stroke as single stroke
512	First approach start position

Note

The option values 0-3 encode the behavior at reversal points at Power OFF. You can choose one of the alternatives 0-3. The other settings can be combined with the selected alternative according to individual requirements. A + character can be inserted to create a string of options.

Example: The oscillation movement for axis Z must stop at reversal point 1 on deactivation; an end position must then be approached and a newly programmed feedrate take immediate effect; the axis must stop immediately after deletion of distance-to-go.

OSCTRL[Z] = (1+4, 16+32+64)

The set/reset options are entered into the appropriate setting data in synchronism with the block in the main run and thus remain effective until the setting data is next changed.

Note

The control evaluates the reset options, then the set options.

7) Sparking-out strokes:

- OSNSC[oscillation axis] = number of sparking-out strokes

The number of sparking-out strokes is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed.

8) End position to be approached after deactivation of oscillation:

- OSE[oscillation axis] = end position of oscillation axis

The end position is entered into the appropriate setting data in synchronism with the block in the main run and thus remains effective until the setting data is next changed. Option value 4 is set implicitly according to Table 7-3, such that the set end position is approached.

9) Start position to be approached prior to activation of oscillation:

- OSB [oscillation axis] = start position of oscillation axis

The start position is entered into the appropriate setting data SD43790 $SA_OSCILL_START_POS in synchronism with the block in the main run and thus remains effective until the setting data is next changed. Bit 9 in setting data SD43770 $SA_OSCILL_CTRL_MASK must be set to initiate an approach to the start position. The start position is approached **before reversal point 1**. If the start position coincides with reversal position 1, reversal position 2 is approached next.

As an alternative to programming command OSB, it is also possible to enter the start position directly in setting data SD43790 $SA_OSCILL_START_POS.

All positional information in the setting data and system variables refer to the basic coordinate system (BCS). The positional data for OSB, OSE refer to the workpiece coordinate system (WCS).

No halt time applies when the start position is reached, even if this position coincides with reversal position 1; instead, the axis waits for the exact stop fine signal. Any configured exact stop condition is fulfilled.

If a non-modal oscillation process does not require an infeed motion if the start position coincides with reversal position 1, this option can be configured with another synchronized action, see examples in the chapter "Non-modal oscillation (starting position = reversal point 1)".

Programming example

The "Examples" chapter gives an example containing all the important elements for asynchronous oscillation.

12.2.2 Asynchronous oscillation under PLC control

Activating

The function can be selected from the PLC via setting data OSCILL_IS_ACTIVE in all operating modes except for MDA Ref and JOG Ref.

Settings

The following criteria can be controlled from the PLC via setting data: Activation and deactivation of oscillation movement, positions of reversal points 1 and 2, stop times at reversal points, feedrate velocity, the options in the reversal points, the number of sparking-out strokes and the end position after deactivation. However, these values can also be set beforehand as a setting data via the HMI user interface directly or via an NC program. These settings remain valid after power ON and the PLC can also start an oscillation movement set in this way directly via setting data OSCILL_IS_ACTIVE (via variable service).

Constraints

A spindle which must act as an axis to execute an oscillation movement started via the PLC must fulfill the conditions required to allow traversal as a positioning axis, i.e. the spindle must, for example, have been switched to the position control (`SPOS`) beforehand.

The two stop bits of the VDI interface IS DB31, ... DBX28.6 ("Stop") and IS DB31, ... DBX28.5 ("Stop at next reversal point") are always interpreted by the axes irrespective of whether the axis is controlled by the PLC or not.

12.2.3 Special reactions during asynchronous oscillation

With PLC control

The PLC program can take over the control of an oscillation axis via VDI signals. These VDI signals also include program end, operating mode changeover and single block.

The following VDI interface signals are ignored in SW 6.2 and earlier: Feed/spindle stop and NC-STOP; the resulting deceleration request is suppressed in the case of delete distance-to-go.

In SW 6.3 and later, an asynchronous reciprocating axis interpolated by the main run reacts to NCSTOP, alarm handling, end of program, program control and RESET.

The PLC controls the axis/spindle via the axial VDI interface (PLC→NCK) by means of the IS DB31, ... DBX28.7 ("PLC controls axis") == 1

For further information about axes with PLC control, please see:

References:

/FB2/ Function Manual, Extended Functions; Positioning Axes (P2)

Without PLC control

If the PLC does not have control over the axis, then the axis is treated like a normal positioning axis (POSA) during asynchronous oscillation.

Delete distance-to-go

Channel-specific delete distance-to-go is ignored. Axial delete distance-to-go:

Without PLC control

If the oscillation axis is not under PLC control, it is stopped by means of a braking ramp.

With PLC control

In this case, deceleration is suppressed and must be initiated by the PLC.

The following applies to **both** cases: After the axis has been stopped, the appropriate reversal point is approached (see OSCILL_CONTROL_MASK, Chapter 4) and the distance-to-go deleted. The sparking-out strokes are then executed and the end position is approached if this has been set as such in OSCILL_CONTROL_MASK.

The oscillation movement is then finished.

Note

During grinding, the calipers can be put into action via axial delete distance-to-go.

EMERGENCY STOP

In the event of an EMERGENCY STOP, the axis is decelerated by the servo (by cancellation of servo enable and follow-up). The oscillation movement is thus terminated and must be restarted if necessary.

Reset

The oscillation movement is interrupted and deselected with a braking ramp. The options selected subsequently are not processed (sparking-out strokes, end point approach).

Working area limitation, limit switches

If it is detected during preprocessing that the oscillation movement would violate an active limitation, then an alarm is output and the oscillation movement not started. If the oscillation axis violates a limitation which has been activated in the meantime (e.g. 2nd software limit switch), then the axis is decelerated down along a ramp and an alarm output.

⚠ CAUTION
Protection zones are not effective!

Follow-up mode

There is no difference to positioning axes.

End of program

If the axis is not controlled by the PLC, then the program end is not reached until the oscillation movement is terminated (reaction as for POSA:

Positioning across block boundaries).

If the axis is controlled by the PLC, then it continues to oscillate after program end.

Mode change

The following table shows the operating modes in which oscillation can be implemented. Changeover to an operating mode which allows oscillation does not affect the oscillation movement. Changeover to inadmissible operating modes is rejected with an alarm. It is not possible to traverse an axis in oscillation mode while applying control commands from the NC program or via operator inputs (JOG) simultaneously; an alarm is output if this is attempted. The following applies: The type of movement first started has priority.

Table 12- 4 Operating modes which allow oscillation

Operating mode	Allows oscillation
AUTO	Yes
MDA	Yes
MDA Repos	Yes
MDA Teachin	Yes
MDA Ref	No
JOG	Yes
JOG Ref	No
JOG Repos	Yes

Single-block processing

If the axis is not controlled by the PLC, then it responds to a single block in the same way as a positioning axis (POSA), i.e. it continues the movement.

Override

The override is specified by the:

VDI interface

Axial override acts on the oscillation axis.

Programming

The override acts on oscillation axes in the same way as on positioning axes.

Block search

In Block Search the last valid oscillation function is registered and the machine data OSCILL_MODE_MASK is activated (default) accordingly, either directly after NC start (when approaching the start position after block search) or after reaching the start position after block search.

OSCILL_MODE_MASK Bit 0:

0: Oscillation starts after reaching the start position.

1: Oscillation starts immediately after NC start.

REORG

Reversal point 1 is always approached first before oscillation continues.

ASUB

The oscillation movement continues while an ASUB (asynchronous subprogram) is in progress.

12.3 Oscillation controlled by synchronized actions

General procedure

An asynchronous oscillation movement is coupled via synchronized actions with an infeed motion and controlled accordingly.

References:

/FBSY/ Function Manual, Synchronized Actions

The following description concentrates solely on the motion-synchronous actions associated with the oscillation function.

Functions

The following function complexes can be implemented by means of the language tools described in detail below:

1. Infeed at the reversal point (see Chapter "Infeed at Reversal Point 1 or 2").
2. Infeed in reversal point range (see Chapter "Infeed in Reversal Point Range").
3. Infeed in both reversal points (see Chapter "Infeed in both Reversal Points").
4. Stopping oscillation movement at reversal point until infeed is terminated (see Chapter "Stopping Oscillation Movement at Reversal Point").
5. Enable oscillation movement (see Chapter "Enable Oscillation Movement").
6. Preventing premature start of partial infeed (see Chapter "Preventing premature start of partial infeed").

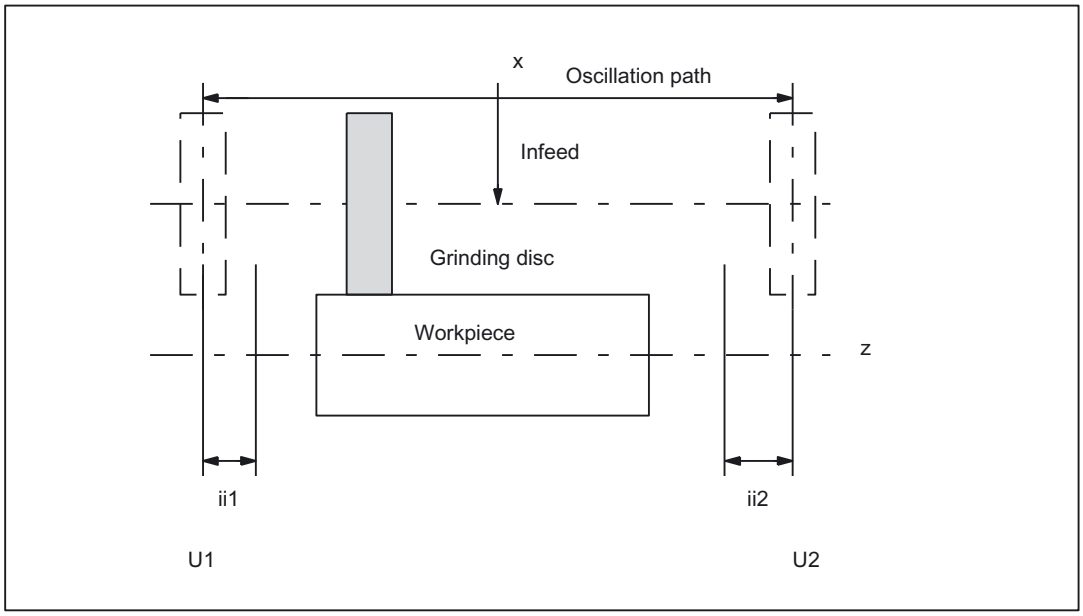


Figure 12-1 Arrangement of oscillation and infeed axes plus terms

Legend:
U1: Reversal point 1
U2: Reversal point 2
ii1: Reversal point range 1
ii2: Reversal point range 2

Programming

The parameters governing oscillation (see Chapter "Assigning Oscillation and Infeed Axis OSCILL") must be defined before the movement block containing the assignment between the infeed and oscillation axes (see), the infeed definition (POSP) and the motion-synchronous actions:

The oscillation axis is enabled via a WAITP [oscillation axis] (see MD30552 $MA_AUTO_GET_TYPE), allowing the oscillation parameters to be transferred, i.e. into the setting data, simultaneously. The symbolic names, e.g. SA43700 $SA_REVERSE_POS1 can then be used to program the motion-synchronous actions.

Note

For motion-synchronous actions with $SA_REVERSE_POS values, the comparison **values at the time of interpretation** are valid. If setting data is changed afterwards, this has no effect.

For motion-synchronous actions with $AA_REVERSE_POS values, the comparison values within the **interpoltation** are valid. This ensures a reaction to the modified reversal positions.

- **Motion-synchronous conditions WHEN, WHENEVER**
- Activation through motion block
  - Assign oscillation axis and infeed axes to one another OSCILL
  - Specify infeed response POSP.

The following sections present those elements which have not yet been dealt with.

Some examples are described in the chapter "Examples".

Note

If the condition with which the motion-synchronous action (WHEN and WHENEVER) has been defined is no longer valid, the OVERRIDE for this condition is **automatically** set to 100% if the OVERRIDE had been set to 0% before.

Main run evaluation

It is possible to compare the synchronization conditions in the interpolation cycle in the main run with the current actual values ($$ variable on the right of comparison conditions). With normal system variable comparison, the expressions are evaluated in the first run. The complete extended possibilities for synchronized actions are listed in the following documentation:

References:

/FBSY/ Function Manual, Synchronized Actions

Example 1

Oscillation, reversal position firmly set via setting data:

```
$SA_OSCILL_REVERSE_POS1[Z]=-10
$SA_OSCILL_REVERSE_POS2[Z]=10
G0 X0 Z0
WAITP(Z)
ID=1 WHENEVER $AA_IM[Z] <
$SA_OSCILL_REVERSE_POS1[Z] DO
$AA_OVR[X]=0
ID=2 WHENEVER $AA_IM[Z] >
$SA_OSCILL_REVERSE_POS2[Z] DO
$AA_OVR[X]=0
                                    ; If the actual value of the oscillation axis
                                    ; has exceeded the reversal point,
                                    ; the infeed axis is stopped.
OS[Z]=1 FA[X]=1000 POS[X]=40        ; Switch on oscillation
OS[Z]=0                             ; Switch off oscillation
M30
```

Example 2

Oscillation with online change of the reversal position, i.e. any modification of reversal position 1 via the user surface are immediately taken into account with active oscillation movement.

```
$SA_OSCILL_REVERSE_POS1[Z]=-10
$SA_OSCILL_REVERSE_POS2[Z]=10
G0 X0 Z0
WAITP(Z)
ID=1 WHENEVER $AA_IM[Z] <
$$SA_OSCILL_REVERSE_POS1[Z] DO
$AA_OVR[X]=0
ID=2 WHENEVER $AA_IM[Z] >
$$SA_OSCILL_REVERSE_POS2[Z] DO
$AA_OVR[X]=0
                                    ; If the actual value of the oscillation axis
                                    ; has exceeded the reversal point,
                                    ; the infeed axis is stopped.
OS[Z]=1 FA[X]=1000 POS[X]=40        ; Switch on oscillation
OS[Z]=0                             ; Switch off oscillation
M30
```

12.3.1 Infeed at reversal point 1 or 2

Function

As long as the oscillation axis has not reached the reversal point, the infeed axis does not move.

Application

Direct infeed in reversal point

Programming

For reversal point 1:

WHENEVER $AA_IM[Z] <> $SA_OSCILL_REVERSE_POS1[Z]

DO $AA_OVR[X] = 0 $AA_OVR[Z] = 100

For reversal point 2:

WHENEVER $AA_IM[Z] <> $SA_OSCILL_REVERSE_POS2[Z]

DO $AA_OVR[X] = 0 $AA_OVR[Z] = 100

Explanation of system variables:

$AA_IM[Z]: Current position of oscillating axis Z in the MCS

$SA_OSCILL_REVERSE_POS1[Z]: Position of the reversal point1 of the oscillation axis

$AA_OVR[X]: Axial override of the infeed axis

$AA_OVR[Z]: Axial override of the oscillation axis

Explanation of key words:

WHENEVER ... DO ... Whenever condition is fulfilled, then ...

Infeed

The absolute infeed value is defined by instruction POSP.

See Chapter "Defining Infeed POSP".

Assignment

The assignment between the oscillation axis and the infeed axis is defined by instruction OSCILL.

See Chapter "Assignment of Oscillating and Infeed Axes OSCILL".

12.3.2 Infeed in reversal point range

Function

Reversal point range 1:

No infeed takes place provided the oscillation axis has not reached the reversal point range (position at reversal point 1 plus contents of variables ii1). This applies on the condition that reversal point 1 is set to a lower value than reversal point 2. If this is not the case, then the condition must be changed accordingly.

Application

Reversal point range 1:

The purpose of the synchronized action is to prevent the infeed movement from starting until the oscillation movement has reached reversal point range 1.

See the figure in Chapter "Controlling Oscillation via Synchronous Actions".

Programming

Reversal point range 1:

WHENEVER $AA_IM[Z] > $SA_OSCILL_REVERSE_POS1[Z] + ii1

DO $AA_OVR[X] = 0

Explanation of system variables:

$AA_IM[Z]: Current position of oscillating axis Z

$SA_OSCILL_REVERSE_POS1[Z]: Position of reversal point 1 of the oscillation axis

$AA_OVR[X]: Axial override of the infeed axis

ii1: Magnitude of reversal range (user Variable)

Explanation of key words:

WHENEVER ... DO ... Whenever condition is fulfilled, then ...

Function

Reversal point range 2:

The infeed axis stops until the current position (value) of the oscillation axis is lower than the position at reversal point2 minus the contents of variable ii2. This applies on condition that the setting for reversal point position 2 is higher than that for reversal point position 1. If this is not the case, then the condition must be changed accordingly.

Application

Reversal point range 2:

The purpose of the synchronized action is to prevent the infeed movement from starting until the oscillation movement has reached reversal point range 2.

See the figure in Chapter "Controlling Oscillation via Synchronous Actions".

Programming

Reversal point range 2:

WHENEVER $AA_IM[Z] < $SA_OSCILL_REVERSE_POS2[Z] - ii2

DO $AA_OVR[X] = 0

Explanation:

$AA_IM[Z]: Current position of oscillating axis Z

$SA_OSCILL_REVERSE_POS2[Z]: Position of reversal point 2 of the oscillation axis

$AA_OVR[X]: Axial override of the infeed axis

ii2: Magnitude of reversal range 2 (user variable)

Infeed

The absolute infeed value is defined by instruction POSP.

See Chapter "Defining Infeed POSP".

Assignment

The assignment between the oscillation axis and the infeed axis is defined by instruction OSCILL.

See Chapter "Assignment of Oscillating and Infeed Axes OSCILL".

12.3.3 Infeed at both reversal points

General procedure

The functions described above for infeed at the reversal point and in the reversal point range can be freely combined.

Combinations

Infeed:

to U1 - to U2

to U1 - range U2

range U1 - to U2

range U1 - range U2

One-sided infeed

to U1

to U2

range U1

range U2

These options are described in the chapter "Infeed in Reversal Point 1 or 2" and the chapter "Infeed in the Reversal Range".

12.3.4 Stop oscillation movement at the reversal point

Function

Reversal point 1:

Every time the oscillation axis reaches reversal position 1, it must be stopped by means of the override and the infeed movement started.

Application

The synchronized action is used to hold the oscillation axis stationary until part infeed has been executed. This synchronized action can be omitted if the oscillation axis need not wait at reversal point 1 until part infeed has been executed. At the same time, this synchronized action can be used to start the infeed movement if this has been stopped by a previous synchronized action which is still active.

Programming

WHENEVER $AA_IM[oscillation axis] == $SA_OSCILL_REVERSE_POS1[oscillation axis]

DO $AA_OVR[oscillation axis] = 0 $AA_OVR[infeed axis] = 100

Explanation of system variables:

$AA_IM[oscillation axis]: Current position of oscillation axis

$SA_OSCILL_REVERSE_POS1[oscillation axis]: Reversal point of the oscillation axis

$AA_OVR[oscillation axis]: Axial override of the oscillation axis

$AA_OVR[infeed axis]: Axial override of the infeed axis

Function

Reversal point 2:

Every time the oscillation axis reaches reversal position 2, it must be stopped by means of the override 0 and the infeed movement started.

Application

The synchronized action is used to hold the oscillation axis stationary until part infeed has been executed. This synchronized action can be omitted if the oscillation axis need not wait at reversal point 2 until part infeed has been executed. At the same time, this synchronized action can be used to start the infeed movement if this has been stopped by a previous synchronized action which is still active.

Programming

WHENEVER $AA_IM[oscillation axis] == $SA_OSCILL_REVERSE_POS2[oscillation axis]

DO $AA_OVR[oscillation axis] = 0 $AA_OVR[infeed axis] = 100

Explanation:

$AA_IM[oscillation axis]: Current position of oscillation axis

$SA_OSCILL_REVERSE_POS2[oscillation axis]: Reversal point 2 of the oscillation axis

$AA_OVR[oscillation axis]: Axial override of the oscillation axis

$AA_OVR[infeed axis]: Axial override of the infeed axis

12.3.5 Oscillation movement restarting

Function

The oscillation axis is started via the override whenever the distance-to-go for the currently traversed path section of the infeed axis = 0, i.e. part infeed has been executed.

Application

The purpose of this synchronized action is to continue the movement of the oscillation axis on completion of the part infeed movement. If the oscillation axis need not wait for completion of partial infeed, then the motion-synchronous action with which the oscillation axis is stopped at the reversal point must be omitted.

Programming

WHENEVER $AA_DTEPW[infeed axis] == 0

DO $AA_OVR[oscillation axis] =100

Explanation of system variables:

$AA_DTEPW[infeed axis]: axial remaining travel distance for the infeed axis in the MCS: Path distance of the infeed axis

$AA_OVR[oscillation axis]: Axial override for oscillation axis

Explanation of key words:

WHENEVER ... DO ... Whenever condition is fulfilled, then ...

12.3.6 Do not start partial infeed too early

Function

The functions described above prevent any infeed movement outside the reversal point or the reversal point range. On completion of an infeed movement, however, restart of the next partial infeed must be prevented.

Application

A channel-specific flag is used for this purpose. This flag is set at the end of the partial infeed (partial distance-to-go == 0) and is deleted when the axis leaves the reversal point range. The next infeed movement is then prevented by a synchronized action.

Programming

WHENEVER $AA_DTEPW[infeed axis] == 0

DO $AC_MARKER[index]=1

and, for instance, for reversal point 1:

WHENEVER $AA_IM[Z] <> $SA_OSCILL_REVERSE_POS1[Z]

DO $AC_MARKER[Index]=0

WHENEVER $AC_MARKER[index]==1

DO $AA_OVR[infeed axis]=0

Explanation of system variables:

$AA_DTEPW[infeed axis]: axial remaining travel distance for the infeed axis in the MCS: Path distance of the infeed axis

$AC_MARKER[index]: Channel-specific marker with index

$AA_IM[oscillation axis]: Current position of oscillation axis

$SA_OSCILL_REVERSE_POS1[oscillation axis]: Reversal point 1 of the oscillation axis

$AA_OVR[infeed axis]: Axial override for infeed axis

Explanation of key words:

WHENEVER ... DO ... Whenever condition is fulfilled, then ...

12.3.7 Assignment of oscillation and infeed axes OSCILL

Function

One or several infeed axes are assigned to the oscillation axis with command OSCILL. Oscillation motion starts.

The PLC is informed of which axes have been assigned via the VDI interface. If the PLC is controlling the oscillation axis, it must now also monitor the infeed axes and use the signals for the infeed axes to generate the reactions on the oscillation axis via 2 stop bits of the interface.

Application

The axes whose response has already been defined by synchronous conditions are assigned to one another for activation of oscillation mode. The oscillation movement is started.

Programming

OSCILL[oscillation axis] = (infeed axis1, infeed axis2, infeed axis3)

Infeed axis2 and infeed axis3 in brackets plus their delimiters can be omitted if they are not required.

12.3.8 Definition of infeeds POSP

Function

The control receives the following data for the infeed axis:

- Total infeed
- Part infeed at reversal point/reversal point range
- Part infeed response at end

Application

This instruction must be given after activation of oscillation with `OSCILL` to inform the controller of the required infeed values at the reversal points/reversal point ranges.

Programming

POSP[infeed axis] = (end position, part section, mode)

End position: End position for the infeed axis after all partial infeeds have been traversed.

Part section: Part infeed at reversal point/reversal point range

Mode 0: For the last two part steps, the remaining path up to the target point is divided into two equally large residual steps (default setting).

Mode 1: The part length is adjusted such that the total of all calculated part lengths corresponds exactly to the path up to the target point.

12.3.9 External oscillation reversal

For example, keys on the PLC can be used to change the oscillation area or instantaneously reverse the direction of oscillation.

The current oscillation motion is braked and the axis then traversed in the opposite direction in response to the edge-triggered PLC input signal **Oscillation reversal** (DB31 DBB28 bit0). The braking operation is checked back via PLC output signal **Oscillation reversal active** (DB31 DBB100 bit 2).

The braking position of the axis can be accepted as the **new reversal position** by means of PLC signal **Change reversal position** (DB31 DBB28 Bit4).

The PLC input signal **Select reversal position** (DB31 DBB28 bit 3) is ignored provided that the change is made in relation to the last issued *External oscillation reversal* command.

No change in the reversal points applied via handwheel or JOG keys may be active for the relevant axis. If handwheel or JOG key changes are currently active, display alarm 20081 (Braking position cannot be accepted as reversal position - handwheel active) will be generated. The alarm is automatically reset when the conflict has been eliminated.

Hold time

No stop time is applied for a change of direction due to an *external oscillation reversal*. The axis waits for the exact stop fine signal. Any configured exact stop condition is fulfilled.

Infeed motion

With non-modal oscillation, no infeed movement is performed for a change of direction due to an *external oscillation reversal* as the reversal position has not been reached and consequently the appropriate synchronized action is not fulfilled.

System variables

The braking position can be scanned via system variable $AA_OSCILL_BREAK_POS1, when approach to reversal position 1 is aborted or via

$AA_OSCILL_BREAK_POS2 when approach to reversal position 2 is aborted.

If the relevant reversal point is approached again, the position of the reversal point can be scanned in $AA_OSCILL_BREAK_POS1 or $AA_OSCILL_BREAK_POS2.

In other words, only after an *External oscillation reversal* command is there a difference between the values in $AA_OSCILL_BREAK_POS1 and $AA_OSCILL_REVERSE_POS1 or the values in $AA_OSCILL_BREAK_POS2 and $AA_OSCILL_REVERSE_POS2.

External oscillation reversal can therefore be **detected** by a synchronized action, see examples.

Special cases

If the PLC input signal "oscillation reversal" is activated as the axis is approaching the start position, the approach movement is aborted and the axis continues by approach interruption position 1.

If the PLC input signal "oscillation reversal" is set during a stop period, the stop timer is deactivated; if exact stop fine has not yet been reached, the axis waits for the exact stop fine reached signal before continuing its motion.

If the PLC input signal "oscillation reversal" is activated as the axis is approaching the end position, the approach movement is aborted and oscillation is terminated.

An example of oscillation reversal can be found in the chapter "Changing Reversal Position with 'External Oscillation Reversal' via Synchronous Actions".

12.4 Marginal conditions

Availability of the "Oscillation" function

Oscillation is an option, available under order number 6FC5 251-0AB04-0AA0.

Asynchronous oscillation and oscillation across blocks is available for NCU570, 571, 572, 573.

Oscillation with motion-synchronous actions is available for NCU 572, 573.

12.5 Examples

Requirements

The examples given below require components of the NC language specified in the sections entitled:

- Asynchronous oscillation

and

- Oscillation controlled by synchronized actions.

12.5.1 Example of asynchronous oscillation

Exercise

The oscillation axis Z must oscillate between -10 and 10. Approach reversal point 1 with exact stop coarse and reversal point 2 without exact stop. The oscillation axis feedrate must be 5000. 3 sparking-out strokes must be executed at the end of the machining operation followed by approach by oscillation axis to end position 30. The feedrate for the infeed axis is 1000, end of the infeed in X direction is at 15.

Program section

```
OSP1[Z]=-10              ; Reversal point 1
OSP2[Z]=10               ; Reversal point 2
OST1[Z]=-1               ; Stop time at reversal point 1: Exact stop coarse
OST2[Z]=-2               ; Stop time at reversal point 2: without exact stop
FA[Z]=5000               ; infeed for oscillating axis
OSNSC[Z]=3               ; three spark-out strokes
OSE[Z]=-3                ; End position
OS1 F500 X15             ; start oscillation, infeed X axis
                         ; with infeed 500, feed target 15
```

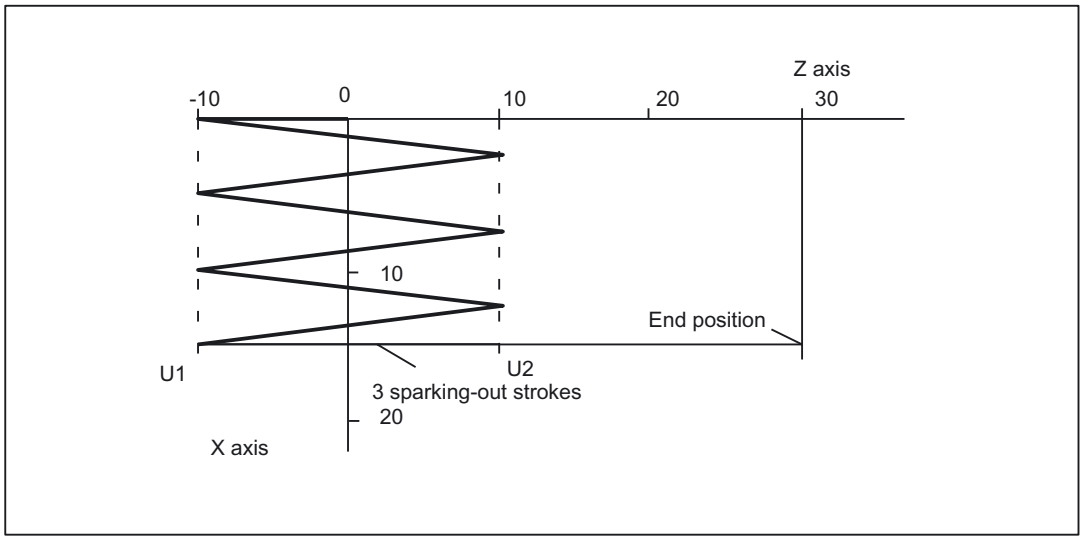


Figure 12-2 Sequences of oscillation movements and infeed, example 1

12.5.2 Example 1 of oscillation with synchronized actions

Exercise

Direct infeed must take place at reversal point 1; the oscillation axis must wait until the part infeed has been executed before it can continue traversal. At reversal point 2, the infeed must take place at a distance of -6 from reversal point 2; the oscillation axis must not wait at this reversal point until part infeed has been executed. Axis Z is the oscillation axis and axis X the infeed axis. (See chapter "Controlling Oscillation via Synchronous Actions").

Note

The setting data OSCILL_REVERSE_POS_1/2 are values in the machine coordinate system; therefore comparison is only suitable with $AA_IM[n].

Program section

```
; Example 1: Oscillation with synchronized actions
OSP1[Z]=10 OSP2[Z]=60      ; explain reversal points 1 and 2
OST1[Z]=-2 OST2[Z]=0       ; Reversal point 1: Without exact stop
                           ; Reversal point 2: Exact stop fine
FA[Z]=5000 FA[X]=250       ; Feed for oscillating axis, feedrate, infeed axis
OSCTRL[Z]=(1+8+16.0)       ; Deactivate oscillating motion at reversal point 1
                           ; after DTG spark-out and approach end position;
                           ; after DTG approach relevant reversal position
OSNSC[Z]=3                 ; 3 sparking-out strokes
```

```
OSE[Z]=0                    ; End position = 0;
WAITP(Z)                    ; enable oscillation for Z axis

; motion-synchronous actions
;
; always, when              the current position of the oscillating axis in the MCS
;                           is
; not equal to              reversal position 1
; then                      set the marker with index 1 to value 0 (reset marker 1)
;
WHENEVER $AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[1]=0
;
; always, when              the current position of the oscillating axis in the MCS
;                           is
; less than                 the start of reversal area 2 (here: reversal point 2 -6),
;
; then                      set the axial override of the infeed axis to 0%.
; and                       set the marker with index 2 to value 0 (reset marker 2)
;
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0 $AC_MARKER[2]=0
;
; always, when              the current position of the oscillating axis in the MCS
;                           is
; equal to                  reversal position 1,
; then                      set the axial override of the oscillation axis to 0%
;
; and                       set the axial override of infeed axis Z to 100% (this
;                           cancels the previous synchronous action).
;
WHENEVER $AA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=0 $AA_OVR[X]=100
;
; always, when              the distance-to-go of the part infeed is
; equal to                  0,
; then                      set the marker with index 2 to value 1
; and                       set the marker with index 1 to value 1
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[2]=1 $AC_MARKER[1]=1
;
; always, when              the flag with index 2 is
; equal to                  1,
; then                      set the axial override of the infeed axis to 0%; this
;                           prevents premature infeed (oscillation axis has not left
;                           reversal point 1 yet).
;
;
WHENEVER $AC_MARKER[2]==1 DO $AA_OVR[X]=0
;
; always, when              the flag with index 1 is
```

```
; equal to                 1,
; then                     set the axial override of the infeed axis to 0%; this
;                          prevents premature infeed (oscillation axis has not left
;                          reversal range 2 yet)
;
; and                      set the axial override of the oscillation axis to 100%
;                          ('Start' oscillation)
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
;
; if the current position of the oscillating axis in the MCS is
; equal to                 reversal position 1,
; then                     set the axial override of the oscillation axis to 100%
;
; and                      set the axial override of infeed axis Z to 0% (this
;                          cancels the second synchronous action once only!).
;
WHEN $AA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=100 $AA_OVR[X]=0
;
;---------------------------------
---------
OSCILL[Z]=(X) POSP[X]=(5,1,1)       ; assign axis X to the oscillation axis Z as
;                                   oscillation axis, which has to infeed up to end
;                                   position 5 in steps of 1 and the sum of all
;                                   partial distances must add up to the end
                                    position
;
M30                                 ; End of program
```

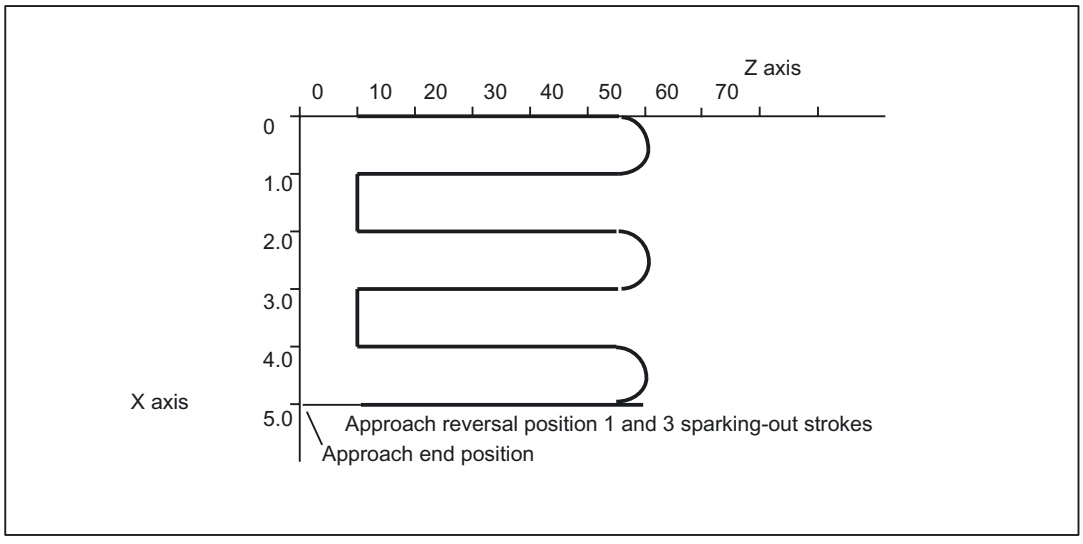


Figure 12-3 Sequences of oscillation movements and infeed, example 1

12.5.3 Example 2 of oscillation with synchronized actions

Exercise

No infeed must take place at reversal point 1. At reversal point 2, the infeed must take place at distance ii2 from reversal point 2; the oscillation axis must wait at this reversal point until part infeed has been executed. Axis Z is the oscillation axis and axis X the infeed axis.

Program section

Example 2: Oscillation with synchronized actions

```
DEF INT ii2                ; Define variable for reversal area 2
;
OSP1[Z]=10 OSP2[Z]=60      ; explain reversal points 1 and 2
OST1[Z]=0 OST2[Z]=0        ; Reversal point 1: Exact stop fine
                           ; Reversal point 2: Exact stop fine
FA[Z]=5000 FA[X]=100       ; Feed for oscillating axis, infeed axis
OSCTRL[Z]=(2+8+16,1)       ; Deactivate oscillating motion at reversal point 2
                           ; after remaining distance spark-out and approach end
                             position
                           ; delete approach after remaining distance
                           ; approach reversal position
OSNSC[Z]=3                 ; 3 sparking-out strokes
OSE[Z]=70                  ; End position = 70
ii2=2                      ; Set reversal point range
WAITP(Z)                   ; enable oscillation for Z axis

; motion-synchronous actions:
; always, when             the current position of the oscillating axis in the MCS
;                          is
; less than                the start of reversal area 2
; then                     set the axial override of infeed axis
;                          to 0%
; and                      set the marker with index 0 to value 0
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]-ii2 DO $AA_OVR[X]=0 $AC_MARKER[0]=0
;
; always, when             the current position of the oscillating axis in the MCS
;                          is
; greater or equal to      reversal position 2
; then                     set the axial override of oscillation axis
;                          to 0%
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0
;
; always, when             the distance-to-go of the part infeed is
; equal to                 0,
; then                     set the marker with index 0 to value 1
```

```
WHENEVER $AA_DTEPW[X] == 0 DO $AC_MARKER[0]=1
;
; always, when             the flag with index 0 is
; equal to                 1,
; then                     set the axial override of infeed axis X to 0% in order to
;                          inhibit premature infeed (oscillating axis has not yet
;                          left reversal area 2 but infeed axis is ready for a new
;                          infeed)
;
; and                      set the axial override of oscillating axis to 100% (this
;                          cancels the 2nd synchronized action).
;
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
;
OSCILL[Z]=(X) POSP[X]=(5,1,1)        ; starting the axes
                                     ; axis x is assigned to oscillation axis Z as
                                     ; infeed axis
                                     ; axis X must go to end position 5 in
                                     ; steps of 1
;
M30
```

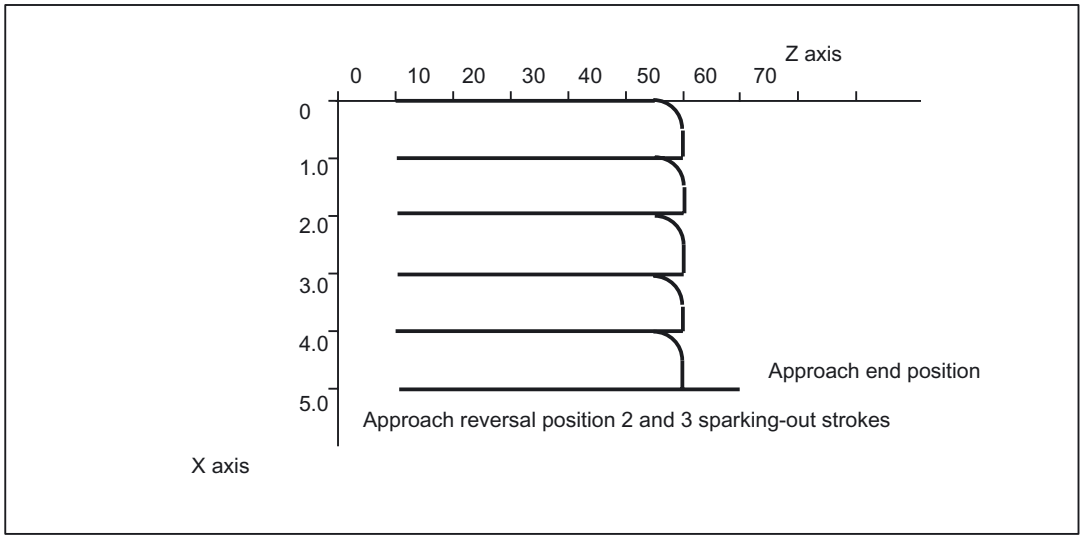


Figure 12-4 Sequences of oscillation movements and infeed, example 2

12.5.4 Examples for starting position

12.5.4.1 Define starting position via language command

```
WAITP(Z)                    ; enable oscillation for Z axis
OSP1[Z]=10 OSP2[Z]=60       ; explain reversal points 1 and 2
OST1[Z]=-2 OST2[Z]=0        ; Reversal point 1: Without exact stop
                            ; Reversal point 2: Exact stop fine
FA[Z]=5000 FA[X]=2000       ; Infeed oscillation axis,
                            ; Feed infeed axis
OSCTRL[Z]=(1+8+16,0)        ; Deactivate oscillating motion at reversal point 1
                            ; after DTG spark-out and approach end position;
                            ; after DTG approach relevant reversal position
OSNSC[Z]=3                  ; 3 sparking-out strokes
OSE[Z]=0                    ; End position = 0
OSB[Z]=0                    ; starting position = 0
OS[Z]=1 X15 F500            ; Start oscillation, continuous infeed
OS[Z]=0                     ; Switch off oscillation
WAITP(Z)                    ; wait for completion of oscillation motion
M30
```

Explanation

When the Z axis starts oscillation, it first approaches the starting position (position = 0 in the example) and then begins the oscillation motion between the reversal points 10 and 60. When the X axis has reached its end position 15, the oscillation finishes with 3 sparking out strokes and approach of end position 0.

12.5.4.2 Start oscillation via setting data

```
WAITP(Z)
STOPRE
$SA_OSCILL_REVERSE_POS1[Z]=-10          ; reversal position 1 = -10
$SA_OSCILL_REVERSE_POS2[Z]=30           ; reversal position 2 = 30
$SA_OSCILL_START_POS[Z] = -50           ; starting position = -50
$SA_OSCILL_CTRL_MASK[Z] = 512           ; Approach starting position,
                                        ; on switch-off stop at the next
                                        ; reversal point
                                        ; do not approach any end position
                                        ; on DTG no sparking-out strokes
$SA_OSCILL_VELO[ Z ] = 5000             ; Infeed for oscillating axis
$SA_OSCILL_IS_ACTIVE[ Z ] = 1           ; Start
$SA_OSCILL_DWELL_TIME1[ Z ] = -2        ; without waiting for exact stop
$SA_OSCILL_DWELL_TIME2[ Z ] = 0         ; wait for fine exact stop
```

```
STOPRE
X30 F100
$SA_OSCILL_IS_ACTIVE[ Z ] = 0              ; stop
WAITP(Z)
M30
```

Description

When the Z axis starts oscillation, it first approaches the starting position (position = -50 in the example) and then begins the oscillation motion between the reversal points -10 and 30. When the X axis has reached its end position 30, the oscillation finishes at the next approached reversal point.

12.5.4.3 Non-modal oscillation (starting position = reversal point 1)

Oscillation with synchronized actions

```
N701                              ; oscillate with synchronous actions,
                                  ; starting position == reversal point 1
;
N702 OSP1[Z]=10 OSP2[Z]=60        ; explain reversal points 1 and 2
N703 OST1[Z]=0 OST2[Z]=0          ; Reversal point 1: Exact stop coarse
                                  ; Reversal point 2: Exact stop fine
N704 FA[Z]=5000 FA[X]=2000        ; Infeed oscillation axis,
                                  ; Feed infeed axis
N705 OSCTRL[Z]=(1+8+16.0)         ; switch off oscillation motion in
                                  ; Reversal point 1 after DTG spark-out
                                  ; and approach end position
                                  ; after DTG relevant reversal position
                                  ; to be approached
N706 OSNSC[Z]=3                   ; 3 sparking-out strokes
N707 OSE[Z]=0                     ; End position = 0
N708 OSB[Z]=10                    ; Starting position = 10
N709 WAITP(Z)                     ; enable oscillation for Z axis
;
; motion-synchronous actions:
; set marker with index 2 on 1 (initialization)
WHEN TRUE DO $AC_MARKER[2]=1
;
always, when                      the marker with index 2 is equal to 0 and the
;                                 current position of the oscillation axis
;                                 is not equal to reversal position 1
Then                              set the marker with index 1 to 0.
;
WHENEVER ($AC_MARKER[2] == 0) AND $AA_IW[Z]>$SA_OSCILL_REVERSE_POS1[Z])
DO $AC_MARKER[1]=0
```

```
; always, when                    the current position of the oscillation axis is
;                                 smaller than the beginning of reversal range 2,
;
; then                            set the axial override of the infeed axis to 0
;                                 and set the marker with index 0 to 0
;
WHENEVER $AA_IW[Z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0 $AC_MARKER[0]=0
;
; always, when                    the current position of the oscillation axis is
;                                 equal to reversal position 1,
; then                            set the axial override of the oscillation axis to
;                                 0 and set the axial override of the infeed axis
;                                 to 100% (this cancels the previous synchronous
;                                 action).
WHENEVER $AA_IW[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=0 $AA_OVR[X]=100
;
; always, when                    the distance-to-go of the partial infeed equals
;                                 0,
; then                            set the marker with index 0 to 1 and set the
;                                 marker with index 1 to 1
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1 $AC_MARKER[1]=1
;
; always, when                    the marker with index 0 equals 1,
; then                            set the axial override of the infeed axis to 0,
;                                 this prevents premature renewed infeed!
;
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0
;
; always, when                    the marker with index 1 equals 1,
; then                            set the axial override of the infeed axis to 0,
;                                 (this prevents premature renewed infeed!) and set
;                                 the axial override of the oscillation axis to
;                                 100% (this cancels the previous synchronous
;                                 action!)
;
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
;
; When                            the current position of the oscillation axis is
;                                 equal to reversal position 1,
; then                            reset the marker with index 2, enable the first
;                                 synchronous action (no infeed when starting
;                                 position is reached == reversal position1)
WHEN $AA_IW[Z]==$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[2]=0
;
;---------------------------------
N750 OSCILL[Z]=(X) POSP[X]=(5,1,1)
; assign axis X to the oscillation axis Z as infeed axis,
; which has to infeed up to end position 5
; in steps of 1 and the sum of all partial distances
```

```
; must add up to the end position.
;
N780 WAITP(Z)                    ; release the Z axis
;
N790 X0 Z0
N799 M30                         ; End of program
```

Description

The starting position matches reversal point 1. The synchronous actions WHEN ... (see above) prevent an infeed when the starting position is reached.

12.5.5 Example of external oscillation reversal

12.5.5.1 Change reversal position via synchronized action with "external oscillation reversal"

```
DEFINE BREAKPZ AS $AA_OSCILL_BREAK_POS1[Z]
DEFINE REVPZ AS $SA_OSCILL_REVERSE_POS1[Z]

WAITP(Z)                       ; enable oscillation for the Z axis
OSP1[Z]=10 OSP2[Z]=60          ; explain reversal points 1 and 2
OSE[Z]=0                       ; End position = 0
OSB[Z]=0                       ; Starting position = 0

                               ; at external reversal of oscillation for
                               ; oscillation reversal point 1, adapt this
WHENEVER BREAKPZ <> REVPZ DO $$SA_OSCILL_REVERSE_POS1 = BREAKPZ
OS[Z]=1 X150 F500              ; Start oscillation, continuous infeed
OS[Z]=0                        ; Switch off oscillation
WAITP(Z)                       ; wait for completion of oscillation motion
M30
```

12.6 Data lists

12.6.1 Machine data

12.6.1.1 General machine data

Number	Identifier: $MN_	Description
10710	PROG_SD_RESET_SAVE_TAB	Oscillations to be saved from SD
11460	OSCILL_MODE_MASK	Control screen form for asynchronous oscillation

12.6.2 Setting data

12.6.2.1 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43700	OSCILL_REVERSE_POS1	Position at reversal point 1
43710	OSCILL_REVERSE_POS2	Position at reversal point 2
43720	OSCILL_DWELL_TIME1	Stop time at reversal point 1
43730	OSCILL_DWELL_TIME2	Stop time at reversal point 2
43740	OSCILL_VELO	Feed velocity of oscillation axis
43750	OSCILL_NUM_SPARK_CYCLES	Number of sparking-out strokes
43760	OSCILL_END_POS	Position after sparking-out strokes/at end of oscillation movement
43770	OSCILL_CTRL_MASK	Control screen form for oscillation
43780	OSCILL_IS_ACTIVE	Oscillation movement ON/OFF
43790	OSCILL_START_POS	Position that is approached after oscillation before reversal point 1, if activated in SD43770:

12.6.3 Signals

12.6.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
External oscillation reversal	DB31, ...DBX28.0	-
Set reversal point	DB31, ...DBX28.3	-
Alter reversal point	DB31, ...DBX28.4	-

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Stop at next reversal point	DB31, ...DBX28.5	-
Stop along braking ramp	DB31, ...DBX28.6	-
PLC-controlled axis	DB31, ...DBX28.7	-

12.6.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Oscillation reversal is active	DB31, ...DBX100.2	-
Oscillation cannot start	DB31, ...DBX100.3	-
Error during oscillation movement	DB31, ...DBX100.4	-
Sparking-out active	DB31, ...DBX100.5	-
Oscillation movement active	DB31, ...DBX100.6	-
Oscillation active	DB31, ...DBX100.7	-

12.6.4 System variables

12.6.4.1 Main run variables for motion-synchronous actions

Main run variable_read

The following variables are provided for main run variable_read:

$A_IN[<arith. Expression>**]**	Digital input (Boolean)
$A_OUT[<arith. Expression>**]**	Digital output (Boolean)
$A_INA[<arith. Expression>**]**	Analog input (Boolean)
$A_OUTA[<arith. Expression>**]**	Analog output (Boolean)
$A_INCO[<arith. Expression>**]**	Comparator inputs (Boolean)
$AA_IW[<axial expression>**]**	Actual position PCS axis (real)
$AA_IB[<axial expression>**]**	Actual position BCS axis (real)
$AA_IM[<axial expression>**]**	Actual position MCS axis (IPO setpoints) (real) With $AA_IM[S1] setpoints for spindles can be evaluated. Modulo calculation is used for spindles and rotary axes, depending on machine data $MA_ROT_IS_MODULO and $MA_DISPLAY_IS_MODULO.
$AA_OSCILL_BREAK_POS1	Breaking position after external oscillation reversal when approaching reversal point 1
$AA_OSCILL_BREAK_POS2	Breaking position after external oscillation reversal when approaching reversal point 2

$AC_TIME	Time from the start of the block (real) in seconds (including the times for the internally generated intermediate blocks)
$AC_TIMES	Time from the start of the block (real) in seconds (without times for the internally generated intermediate blocks)
$AC_TIMEC	Time from the start of the block (real) in IPO steps (including steps for the internally generated intermediate blocks)
$AC_TIMESC	Time from the start of the block (real) in IPO steps (without steps for the internally generated intermediate blocks)
$AC_DTBB	Distance from beginning of block in BCS (Distance to begin, baseCoor) (real)
$AC_DTBW	Distance from beginning of block in PCS (Distance to begin, workpieceCoor) (real)
$AA_DTBB[<axial expression>**]**	Axial distance from beginning of block in BCS (Distance to begin, baseCoor) (real)
$AA_DTBW[<axial expression>**]**	Axial distance from beginning of block in PCS (Distance to begin, workpieceCoor) (real)
$AC_DTEB	Distance to end of block in BCS (Distance to end) (Distance to end, baseCoor) (real)
$AC_DTEW	Distance to end of block in PCS (Distance to end, workpieceCoor) (real)
$AA_DTEB[<axial expression>**]**	Axial distance to end of movement in BCS (Distance to end, baseCoor) (real)
$AA_DTEW[<axial expression>**]**	Axial distance to end of movement in PCS (Distance to end, workpieceCoor) (real)
$AC_PLTBB	Distance from beginning of block in BCS (Path Length from begin, baseCoor) (real)
$AC_PLTEB	Distance to end of block in BCS (Distance to end) (Path Length to end, baseCoor) (real)
$AC_VACTB	Path speed in BCS (Velocity actual, baseCoor) (real)
$AC_VACTW	Path speed in PCS (Velocity actual, workPieceCoor) (real)
$AA_VACTB[<axial expression>**]**	Axis velocity in BCS (Velocity actual, baseCoor) (real)
$AA_VACTW[<axial expression>**]**	Axis velocity in PCS (Velocity actual, workPieceCoor) (real)
$AA_DTEPB[<axial expression>**]**	Axial distance-to-go for oscillation infeed in BCS (Distance to end, pendulum, baseCoor) (real)
$AA_DTEPW[<axial expression>**]**	Axial distance-to-go for oscillation infeed in PCS (Distance to end, pendulum, workpieceCoor) (real)
$AC_DTEPB	Path distance-to-go for oscillation infeed in BCS (not P2) (Distance to end, pendulum, baseCoor) (real)
$AC_DTEPW	Path distance-to-go for oscillation infeed in PCS (not P2) (Distance to end, pendulum, workpieceCoor) (real)

$AC_PATHN	(Path parameter normalized) (real) Normalized path parameter: 0 for beginning of block to 1 for end of block
$AA_LOAD[<axial expression>**]**	Drive utilization (for 611D only)
$AA_POWER[<axial expression>**]**	Drive effective power in W (for 611D only)
$AA_TORQUE[<axial expression>**]**	Drive torque setpoint in Nm (for 611D only)
$AA_CURR[<axial expression>**]**	Actual current value of the axis (for 611D only)
$AC_MARKER[<arithmetic_expression>**]** (int)	Flag variables: can be used to build complex conditions in synchronous actions: 8 Markers (Index 0 - 7) are available. Reset sets the markers to 0. Example: WHEN .....DO $AC_MARKER[0]=2 WHEN .....DO $AC_MARKER[0]=3 WHEN $AC_MARKER[0]==3 DO $AC_OVR=50 Can be read and written independently of synchronous actions in the parts program: IF $AC_MARKER == 4 GOTOF SPRUNG
$AC_PARAM[<arithmetic_expression>**]** (real)	Floating-decimal parameter for synchronous actions. Serves as intermediate saving and evaluation in synchronous actions. 50 Parameters (Index 0 - 49) are available.
$AA_OSCILL_REVERSE_POS1 [<axial expression>**]** (real)	
$AA_OSCILL_REVERSE_POS2 [<axial expression>**]** (real)	Current oscillation reversal points 1 and 2: The current setting data from $SA_OSCILL_REVERSE_POS1 or $SA_OSCILL_REVERSE_POS2 is read. This enables setting data changes at the reversal positions during active oscillation, i.e. during an active synchronous action.

Conditions

Conditions for motion-synchronous actions are formulated:

Main run variable comparison operator expression

For details see:
References:
/FBSY/ Function Manual, Synchronized Actions

R2: Rotary axes

13

13.1 Brief description

Rotary axes in machine tools

Rotary axes are used on many modern machine tools. They are required for tool and workpiece orientation, auxiliary movements and various other technological or kinematic purposes.

A typical example of an application using rotary axes is the 5-axis milling machine. Only with the aid of rotary axes can the tip of the tool be positioned on any point on the workpiece on this type of machine.

Depending on the type of machine, many different demands are placed on a rotary axis. In order that the control can be adapted to the various types of machine, the individual rotary axis functions can be activated by means of machine data or special programming.

Rotary axes are always programmed in degrees. They are generally characterized by the fact that they assume the same position after exactly one rotation (modulo 360 degrees). Depending on the application in question, the traversing range of the rotary axis can be limited to less than 360 degrees (e.g., on swiveling axes for tool holders) or may be unlimited (e.g., when the tool or workpiece is rotated).

In many ways, the responses and features of rotary axes are identical to those of linear axes. The following description of functions is limited to a description of the special features of rotary axes and how they differ from linear axes.

Definition of a rotary axis

An axis can be declared as a rotary axis using the following axis-specific machine data MD30300 $MA_IS_ROT_AX.

Geometry axes are defined as linear axes. Any attempt to define them as rotary axes will be rejected with alarm 4200 (Geometry axis cannot be defined as rotary axis).

Only when an axis has been declared as a rotary axis can it perform or use the functions described on the following pages (e.g., unlimited traversing range, modulo display of axis position, etc.).

Several axes can be declared as rotary axes simultaneously.

Types of rotary axis

Depending on the application, the operating range of a rotary axis can be unlimited (endlessly rotating in both directions [MD30310 $MA_ROT_IS_MODULE = 1]), limited by a software limit switch (e.g., operating range between 0° and 60°) or limited to an appropriate number of rotations (e.g., 1000°).

Some typical rotary-axis applications are listed below.

Typical applications

- 5-axis machining (operating range limited or unlimited)
- Rotary axis for eccentric machining (unlimited operating range)
- Rotary axis for cylindrical or form grinding (unlimited operating range)
- C axis with `TRANSMIT` (unlimited operating range)
- Rotary axis on winding machines (unlimited operating range)
- Rotary workpiece axis (C) on hobbing machines (unlimited operating range)
- Round tool magazines and tool turrets (unlimited operating range)
- Rotary axis for peripheral surface transformation (limited operating range)
- Swivel axes for gripping (operating range 360°)
- Rotary axes for swiveling (operating range < 360°; e.g., 60°)
- Milling swivel axis (A) on hobbing machines (operating range, e.g., 90°)

Axis addresses

Coordinate axes and directions of movement of numerically-controlled machine tools are designated according to DIN.

DIN 66025 specifies the following axis addresses for rotary or swivel axes:

- A, B and C with X, Y and Z as middle axis

This means that A rotates about X, B rotates about Y and C rotates about Z (see fig.).

- The positive rotary-axis direction of rotation corresponds to a clockwise rotation when looking in the positive axis direction of the corresponding middle axis (see fig.).

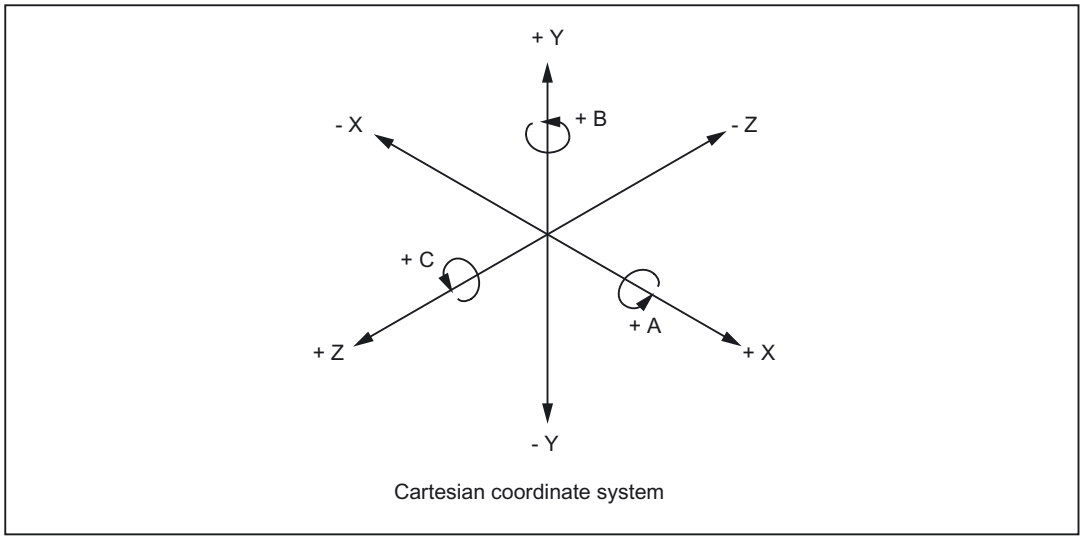


Figure 13-1 Axis identifiers and directions of movement for rotary axes

Extended addressing (e.g., C2=) or freely configured axis addresses can be used for additional rotary axes.

Note

Machine data
MD20050 $MC_AXCONF_GEOAX_ASSIGN_TAB
(assignment of geometry axis to channel axis)
must be adapted to suit the corresponding axis.

Units of measurement

The following units of measurement apply as standard to inputs and outputs for rotary axes:

Units of measurement for rotary axes	
Physical quantity	Unit
Angular position	Degrees
Programmed angular velocity	Degrees/min
MD for angular velocity	[1] rev/min
MD for angular acceleration	[1] rev/sec^2
MD for angular jerk limitation	[1] rev/sec^3

[1] In the case of axis-specific machine data, these units are interpreted by the control as soon as the axis is declared as a rotary axis. The user can define other units for data inputs/outputs using machine data.

References:
/FB1/ Function Manual Basic Functions; Speeds, Setpoint/Actual Value Systems, Control Loop Control (G2)

Operating range

The operating range can be defined by means of axis-specific machine and setting data (software limit switches and working-area limitations). As soon as modulo conversion is activated for the rotary axis (MD30310 $MA ROT_IS_MODULO = 1), the operating range is set to unlimited and the software limit switches and working-area limitations become inactive.

Software limits switches/working-area limitations can also be activated for modulo rotary axes dynamically via the PLC (triggered from the part program via M/H functions, if necessary) using interface signal
DB31, ... DBX12.4 (modulo limit enabled).
The NC acknowledges this activation via interface signal
DB31, ... DBX74.4 (modulo-limit-enabled active).

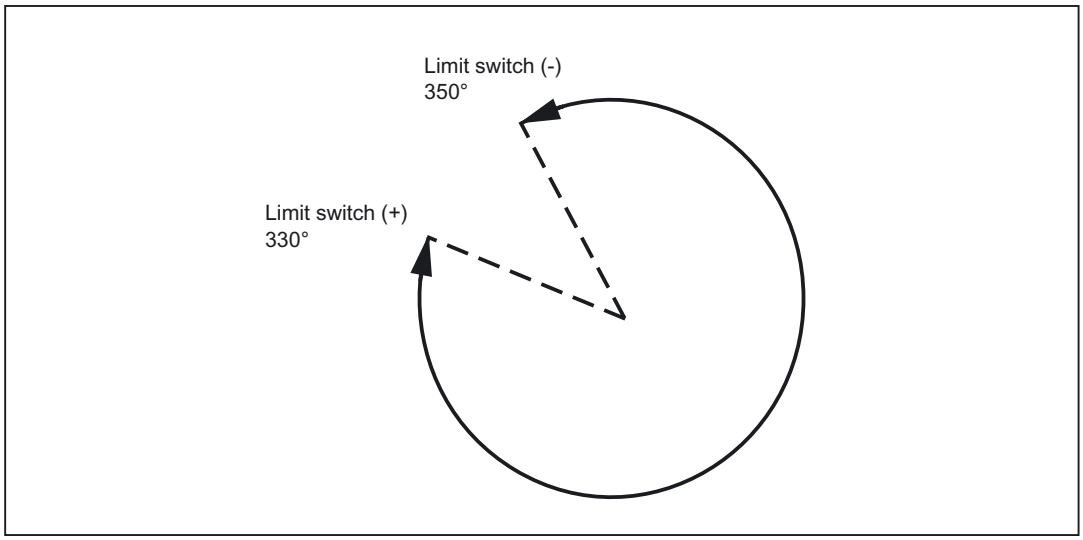


Figure 13-2 Limited operating area of a modulo rotary axis

Position display

The position-display range of values can be set to modulo 360° representation, which is frequently selected for rotary axes, using
MD30320 $MA_DISPLAY_IS_MODULO = 1.

Feedrate

The programmed feedrate F corresponds to an angular velocity (degrees/min) in the case of rotary axes.

If rotary axes and linear axes traverse a common path with G94 or G95, the feedrate should be interpreted in the linear-axis unit of measurement (e.g., mm/min, inch/min).

The tangential velocity of the rotary axis refers to diameter D_{unit} (unit diameter $D_{unit}=360/\pi$). In the case of unit diameter $D=D_{unit}$, the programmed angular velocity in degrees/min and the tangential velocity in mm/min (or inch/min) are numerically identical.

In general, the following applies for tangential velocity:

$F = F_{angle} * D/D_{unit}$	F	= Tangential velocity [mm/min]
	F_{angle}	= Angular velocity [degrees/min]
	D	= Diameter acted on by F [mm]
With $D_{unit} = 360/\pi$	D_{unit}	= Unit diameter [mm]
	π	= Circle constant Pi

Revolutional feedrate

In JOG mode the axis/spindle response also depends on the setting in setting data: SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feedrate active in JOG)

SD41100 $SN_JOG_REV_IS_ACTIVE	
Active	An axis/spindle is always traversed with revolutional feedrate MD32050 $MA_JOG_REV_VELO (revolutional feedrate for JOG) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feedrate for JOG with rapid traverse override), depending on the master spindle.
Not active	The axis/spindle response depends on setting data SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (revolutional feedrate for positioning axes/spindles).
	The response of a geometry axis on which a frame with rotation acts depends on channel-specific setting data SD42600 $SC_JOG_FEED_PER_REV_SOURCE (in JOG mode, revolutional feedrate for geometry axes on which a frame with rotation acts).

13.2 Modulo 360 degrees

Term "modulo 360°"

Rotary axes are frequently programmed in the 360° representation mode. The axis must be defined as a rotary axis in order to use the modulo feature.

With respect to a rotary axis, the term "modulo" refers to the mapping of the axis position within the control in the range 0° to 359.999°. With path defaults > 360° (e.g., for incremental programming using `G91`), the position is mapped in the range of values 0° to 360° following conversion within the control. Mapping is performed in JOG and AUTOMATIC modes. Exception: service display.

In the figure below, the rotary-axis absolute position in the positive direction of rotation is represented as a spiral. An arrow marks the actual absolute position on this spiral (example: point C' = 420°). By tracing the arrow back around the circle (position 0° of the spiral and circle are identical), it is possible to assign an appropriate modulo position within the 360° range to every absolute position. In the example below, absolute position point C' = 420° is mapped onto point C = 60° using modulo conversion.

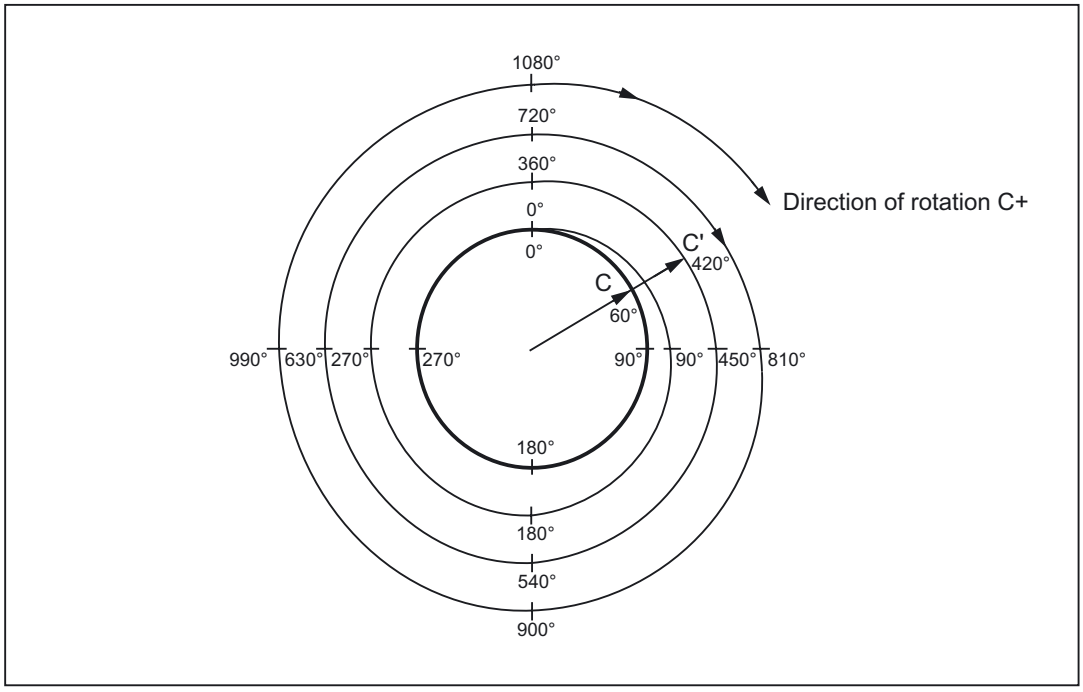


Figure 13-3 Modulo 360° map

Machine-data settings

Machine data can be used to define programming and positioning (MD30310 $MA_ROT_IS_MODULO) as well as the position display (MD30320 $MA_DISPLAY_IS_MODULO) individually in modulo 360° for each rotary axis, depending on the particular machine requirements.

Axis is modulo

MD30310 $MA_ROT_IS_MODULO = 1:

Activation of this machine data allows the special rotary-axis response to be utilized. The rotary-axis positioning response is thus defined during programming (G90, AC, ACP, ACN or DC). A modulo 360° representation is executed within the control after the current work offsets have been taken into account. The resulting **target position within a revolution** is then approached.

The software limit switches and working-area limitations are inactive, meaning that the operating range is **unlimited** (continuously-turning rotary axis).

Modulo position display

MD30320 $MA_DISPLAY_IS_MODULO	
= 1	For rotary axes, a position display with "modulo 360°"(one revolution) is often required, i.e., with a positive direction of rotation the display is periodically reset within the control to 0.000° after 359.999° is reached, with a negative direction of rotation the positions are also displayed in the range 0° to 359.999°.
= 0	Absolute-position display would, in the case of a positive direction of rotation, for example, result in +360° being displayed after one revolution, +720° after two revolutions, etc, in contrast to the modulo 360° display. In this case, the display range is limited by the control in accordance with the linear axes.

MD30320 $MA_DISPLAY_IS_MODULO = 1

Note

The modulo 360° position display should always be selected for a modulo axis (MD30310 $MA_ROT_IS_MODULO = 1).

Starting position for the modulo rotary axis

A start position, not equal to zero, can be specified in machine data for the modulo range: MD30340 $MA_MODULO_RANGE_START.
This allows, for instance, a modulo range of -180° to +180° with a presetting of -180 in MD30340.

The default setting of 0 (degrees) defines a modulo range of 0° - 360°.

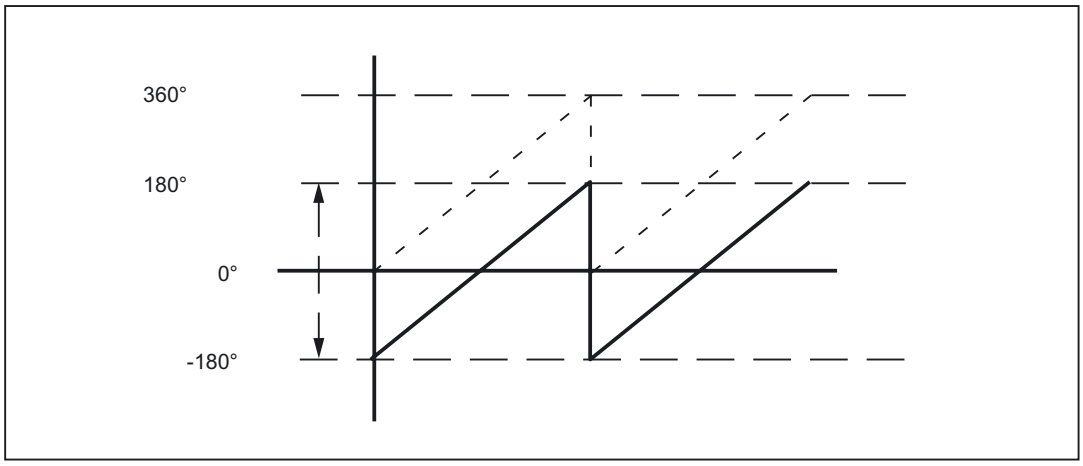


Figure 13-4 Starting position of -180° changes the modulo range to -180° to + 180°

Application

By adjusting the following two machine data
MD30503 $MA_INDEX_AX_OFFSET
MD30340 $MA_MODULO_RANGE_START,
indexing positions of modulo indexing axes can be implemented in the same way as for the modulo range.

References:
/FB2/ Function Manual Extended Functions; Indexing Axes (T1)

13.3 Programming rotary axes

13.3.1 General information

Note

For general information on programming, please refer to:
References:
/PG/ Programming Manual Fundamentals

MD30310

Axis-specific machine data
MD30310 ROT_IS_MODULO (modulo conversion for rotary axis)
is used to define whether the rotary axis behaves as a linear axis during programming and positioning or whether rotary-axis special features are taken into account.

These features and any differences (mainly with respect to absolute programming) are explained on the following pages.

13.3.2 Rotary axis with active modulo conversion (continuously-turning rotary axis)

Activate modulo conversion

→ Set MD30310 $MA_ROT_IS_MODULO = 1.

Note

With modulo axes, it is advisable to set the position display to modulo 360º (set MD30320 $MA_DISPLAY_IS_MODULO = 1).

Absolute programming (AC, ACP, ACN, G90)

Example for positioning axis: **POS[axis name] = ACP(value)**

- The value identifies the rotary-axis target position in a range from 0° to 359.999°.

  Negative values are also possible, if the range is shifted using the following machine data:
  MD30340 $MA_ MODULO_RANGE_START

  and

  MD30330 MA_MODULO_RANGE

- `ACP` (positive) and `ACN` (negative) unambiguously define the rotary-axis traversing direction (irrespective of the actual position).
- When programming `AC` exclusively or with `G90`, the traversing direction depends on the rotary-axis actual position. If the target position is greater than the actual position, the axis traverses in the positive direction, otherwise it traverses in the negative direction.

  The positioning response can be configured using machine data
  MD30455 $MA_MISC_FUNCTION_MASK bit 2:

  Bit 2 = 0: with `G90`, modulo axis positioned as standard using `AC`

  Bit 2 = 1: with `G90`, modulo axis positioned as standard using `DC` (shortest path)
- Use of ACP and ACN: With asymmetrical workpieces, it must be possible to define the traversing direction in order to prevent collisions during rotation.

Example:

C starting position is 0° (see figure below).

①	`POS[C] = ACP(100)`	Rotary axis C traverses to position 100° in the positive direction of rotation
②	`POS[C] = ACN(300)`	C traverses to position 300° in the negative direction of rotation
③	`POS[C] = ACP(240)`	C traverses to position 240° in the positive direction of rotation
④	`POS[C] = AC (0)`	C traverses to position 0° in the negative direction of rotation

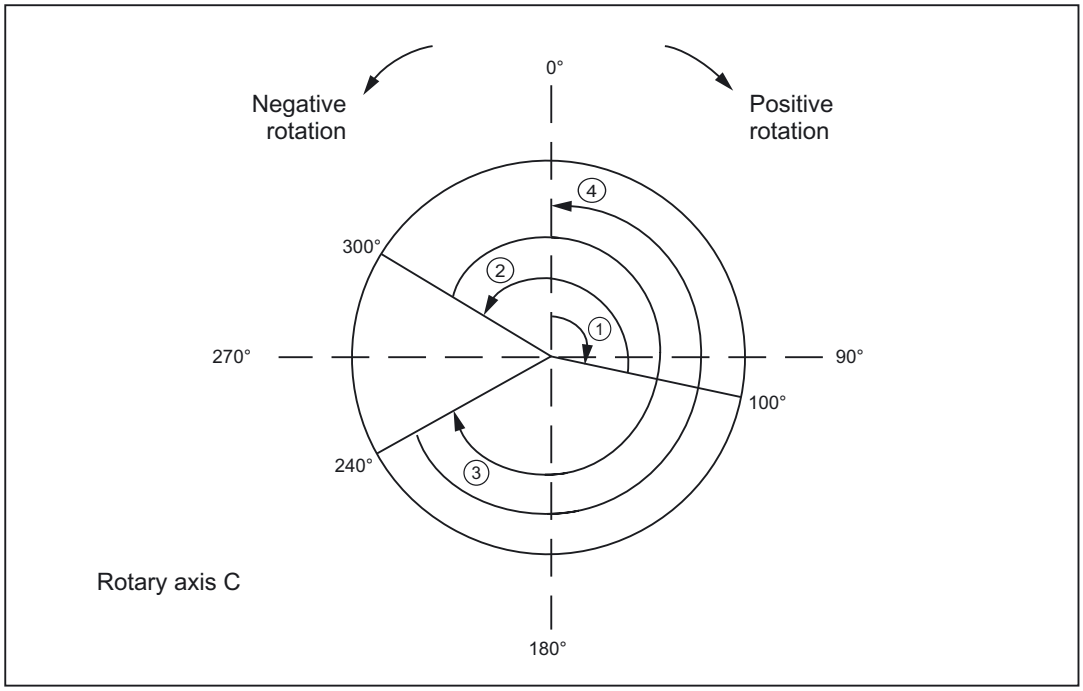


Figure 13-5 Examples of absolute programming for modulo axes

Absolute programming along the shortest path (DC)

POS[axis name] = DC(value)

- The value identifies the rotary-axis target position in a range from 0° to 359.999°. Alarm 16830, "Incorrect modulo position programmed", is output for values with a negative sign or ≥ 360°.
- With DC (Direct Control), the rotary axis approaches the programmed absolute position within one revolution along the **shortest path** (traversing movement max. ∓ 180°).
- The control calculates the direction of rotation and the traverse path according to the current actual position. If the path to be traversed is the same in both directions (180°), the positive direction of rotation receives preference.
- DC application example: the rotary table is required to approach the changeover position in the shortest time (and, therefore, via the shortest path) possible.
- If DC is programmed with a linear axis, alarm 16800, "DC traverse instruction cannot be used", is output.

Example:

C starting position is 0° (see figure below).

①	`POS[C] = DC(100)`	C axis traverses to position 100° along the shortest path
②	`POS[C] = DC(300)`	C axis traverses to position 300° along the shortest path
③	`POS[C] = DC(240)`	C axis traverses to position 240° along the shortest path
④	`POS[C] = DC(60)`	C axis traverses to position 60° along the shortest path. Since, in this case, the path is equal to 180° in both directions, preference is given to the positive direction of rotation.

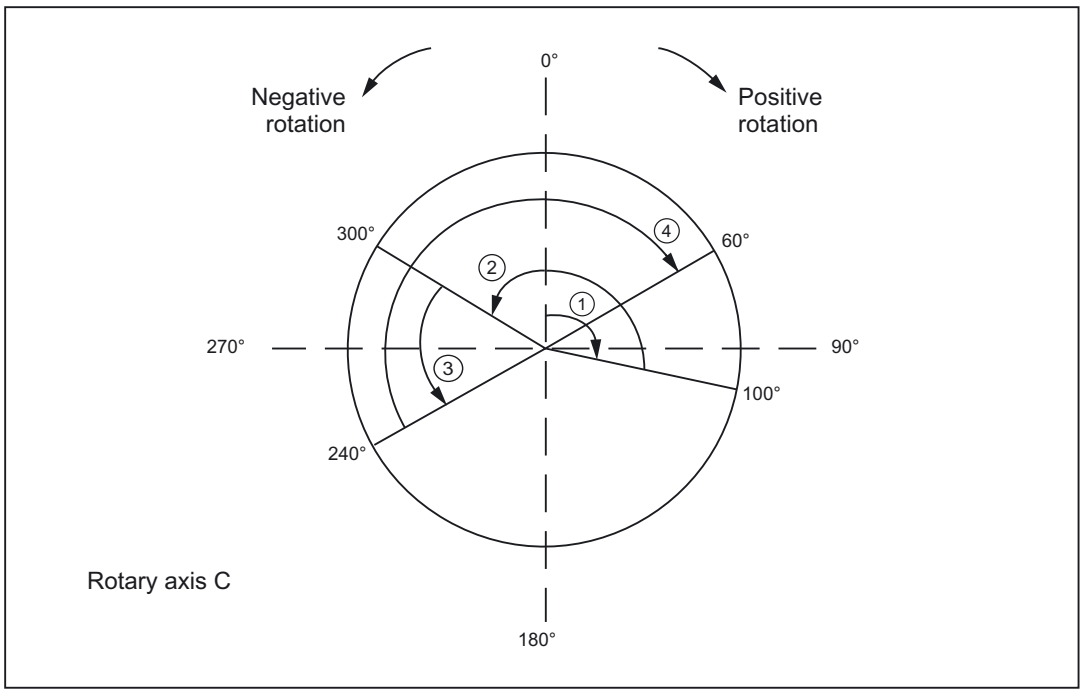


Figure 13-6 Examples of DC programming

Block-search response

After a block search with calculation, the collected modulo-conversion search position can be scanned via the
$AC_RETPOINT
system variable.

This system variable returns the position converted to modulo.

Supplementary conditions for ASUB after block search with calculation:

In this instance, as well as with the cross-channel block-search tool SERUPRO, the modulo conversion simulated in the block search must be performed in the part program.

Modulo rotary axis with/without working-area limitation

The PLC can set interface signal
DB31, ... DBX12.4
for dynamic activation/deactivation of the working-area limitation/software limit switches associated with a modulo rotary axis (similar to rotary axes).

The NC feeds back the current status of the traversing-range limitation via interface signal DB31, ... DBX74.4.

The monitoring function is activated if interface signal
DB31, ... ; DBX 12.4
has been set by the PLC.
The M/H function which causes the PLC to set the interface signal must be followed by `STOPRE`, to ensure through synchronization that only the blocks after switchover are monitored.

Supplementary conditions:

It is only possible to activate/deactivate software-limit-switch monitoring via the PLC interface for modulo axes.

Traversing-range monitoring for modulo axes can be implemented only if the axis is referenced and one limiting pair is active.

This always applies in the case of software limit switches, since these are always activated/deactivated in pairs. To monitor working-area limitations correctly, **both** limitations must have been activated, either via `G26`/`G25` or setting data
SD43400 $SA_ WORKAREA_PLUS_ENABLE

and

SD43410 $SA_WORKAREA_MINUS_ENABLE.

Example of a traversing-range-limitation switchover

A pallet with several clamped workpieces should be machined successively on a modulo rotary axis. This pallet is then replaced by one with a built-on axis whose operating range must be monitored to prevent damage to supply lines.

Configuration:

MD30300 $MA_IS_ROT_AX[AX4] = 1

MD30310 $MA_ROT_IS_MODULO[AX4] = 1

MD36110 $MA_POS_LIMIT_PLUS[AX4] = 340

MD36100 $MA_POS_LIMIT_MINUS[AX4] = 350

Extract from part program:

```
M123                                          ;  Insert the pallet with quadruple
                                                 clamping into the machine
                                                 Deactivate the software limit
                                                 switches on the B axis from the PLC
                                                 DB35, DBX12.4=0
STOPRE                                        ;  Trigger a preprocessing stop
S1000 M3
G4 F2
G1 X0 Y300 Z500 B0 F5000
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle
Z500
B90
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle
Z500
B180
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle
Z500
B270
CYCLE84(500,400,0,350,0,1,4,10,,0,500,1000) ; drilling cycle
Z500
G0 Z540 B0
M124                                          ;  Insert the pallet with built-on
                                                 axis into the machine
                                              ;  Activate the software limit
                                                 switches on the B axis from the PLC
                                                 DB35, DBX12.4=1

STOPRE                                        ;  Trigger a preprocessing stop
B270
```

Incremental programming (IC, G91)

Example for positioning axis: **POS[axis name] = IC(+/-value)**

- The value identifies the rotary-axis traversing distance. The value can be negative and ≥ +/-360°.
- The value's **sign** unequivocally defines the rotary-axis **traversing direction**.
- Application example: milling a spiral groove across several revolutions

Example:

Programming	Effect
`POS[C] = IC(720)`	C axis traverses to 720° incrementally in the positive direction (two revolutions)
`POS[C] = IC(-180)`	C axis traverses to 180° incrementally in the negative direction

Endless traversing range

As soon as the modulo function is active, no limit is placed on the traversing range (software limit switches are not active). The rotary axis can now be programmed to traverse continuously.

Example:

```
LOOP:
POS[C] = IC(720)
GOTOB LOOP
```

13.3.3 Rotary axis without modulo conversion

Deactivate modulo conversion

→ Set MD30310 $MA_ROT_IS_MODULO = 0.

Absolute programming (AC, G90)

Example for positioning axis: **POS[axis name] = AC (+/-value)**

- The value and its sign uniquely identify the rotary-axis target position. The value can be ≥ +/-360°. The position value is limited by the software-limit-switch positions.
- The traversing direction is ascertained by the control according to the signed rotary-axis actual position.
- If `ACP` or `ACN` is programmed, alarm 16810, "ACP traverse instruction cannot be used", or alarm 16820 "ACN traverse instruction cannot be used", is output.
- Application example:

  Linear movements (cam gear) are linked to the rotary axis, thus certain end positions may not be overtraveled.

Example:

Programming	Effect
POS[C] = AC (-100)	Rotary axis C traverses to position -100°; traversing direction depends on the starting position
POS[C] = AC (1500)	Rotary axis C traverses to position 1500°

Absolute programming along the shortest path (DC)

POS[axis name] = DC(value)

Even if the rotary axis is not defined as a modulo axis, the axis can still be positioned with DC (Direct Control). The response is the same as on a modulo axis.

- The value identifies the rotary-axis target position **in a range from 0° to 359.999° (modulo 360°)**. Alarm 16830, "Incorrect modulo position programmed", is output for values with a negative sign or ≥ 360º.
- With DC (Direct Control), the rotary axis approaches the programmed absolute position within one revolution along the **shortest path** (traversing movement max. ± 180°).
- The control calculates the direction of rotation and the traverse path according to the current actual position (in relation to modulo 360°). If the path to be traversed is the same in both directions (180°), the positive direction of rotation receives preference.
- DC application example: the rotary table is required to approach the changeover position in the shortest time (and, therefore, via the shortest path) possible.
- If DC is programmed with a linear axis, alarm 16800, "DC traverse instruction cannot be used", is output.

Example:

Programming	Effect
POS[C] = AC (7200)	Rotary axis C traverses to position 7200°; traversing direction depends on the starting position
POS[C] = DC(300)	Rotary axis C approaches "modulo" position 300° along the shortest path Thus, C traverses about 60° with a negative direction of rotation and stops at absolute position 7140°.
POS[C] = AC (7000)	Rotary axis C traverses to position 7000° absolutely, so C traverses about 140° with a negative direction of rotation

Note

In this example, it would be advisable to activate the modulo 360º display (MD30320 $MA_DISPLAY_IS_MODULO = 1).

Incremental programming (IC, G91)

Example for positioning axis: **POS[axis name] = IC(+/-value)**

When programming with incremental dimensions, the rotary axis traverses across the same path as with the modulo axis. In this case, however, the traversing range is limited by the software limit switches.

- The value identifies the rotary-axis traversing distance.

  The value can be negative and ≥ +/-360°.

- The value's **sign** unequivocally defines the rotary-axis **traversing direction**.

Limited traversing range

The traversing range is limited as with linear axes. The range limits are defined by the "plus" and "minus" software limit switches.

13.3.4 Other programming features relating to rotary axes

Offsets

`TRANS` (absolute) and `ATRANS` (additive) can be applied to rotary axes.

Scalings

`SCALE` or `ASCALE` are not suitable for rotary axes, since the control always bases its modulo calculation on a 360º full circle.

Preset actual-value memory

`PRESETON` is possible.

Indexing axes

References:
/FB2/ Function Manual Extended Functions; Indexing Axes (T1)

13.4 Activating rotary axes

Procedure

The procedure for activating rotary axes is the same as that for linear axes with a small number of exceptions. It should be noted that, as soon as the axis is defined as a rotary axis (MD30300 $MA_IS_ROT_AX = 1), the axis-specific-machine-/setting-data units are interpreted by the control as follows:

Positions	In "degrees"
Velocities	In "rev/min"
Accelerations	In "rev/sec^2"
Jerk limitation	In "rev/sec^3"

Special machine data

Special rotary-axis machine data may also have to be entered, depending on the application:

MD30310 $MA_ROT_IS_MODULO	Modulo conversion for positioning and programming
MD30320 $MA_DISPLAY_IS_MODULO	Modulo conversion for position display
MD10210 $MN_INT_INCR_PER_DEG	Computational resolution for angular positions

The following overview lists the possible combinations of these machine data for a rotary axis:

Possible combinations of rotary-axis machine data				
MD30300	**MD30310**	**MD30320**	**Application permitted**	**Remarks**
0	0	0	Yes	The axis is a linear axis (default).
1	0	0	Yes	The axis is a rotary axis; modulo conversion is not used for positioning, i.e., the software limit switches are active; the position display is absolute.
1	0	1	Yes	The axis is a rotary axis; modulo conversion is not used for positioning, i.e., the software limit switches are active; the position display is modulo; Application: for axes with an operating range of +/-1000°, for example

Possible combinations of rotary-axis machine data				
1	1	1	Yes	The axis is a rotary axis; positioning is performed with modulo conversion, i.e., the software limit switches are inactive, the operating range is unlimited; the position display is modulo (setting most frequently used for rotary axes); axis with/without working-area limitation can be used.
1	1	0	Yes	The axis is a rotary axis; positioning is performed with modulo conversion, i.e., the software limit switches are inactive, the operating range is unlimited; the position display is absolute; axis with/without working-area limitation can be used.
0	0 or 1	0 or 1	Not recommended	Axis is not a rotary axis; therefore, the other MD are not evaluated.

JOG velocity for rotary axes

SD41130 $SN_JOG_ROT_AX_SET_VELO (JOG speed for rotary axes)

The above setting data can be used to specify a JOG velocity valid for all rotary axes.

If a value of 0 is entered in the setting data, axial machine data
MD21150 $MC_JOG_VELO (JOG axis velocity) acts as the rotary-axis JOG velocity.

References:
/FB2/ Function Manual Extended Functions; Manual and Handwheel Travel (H1)

13.5 Special features of rotary axes

Software limit switch

The software limit switches and working-area limitations are active and are required for swivel axes with a limited operating range. However, in the case of continuously-turning rotary axes (MD30310 $MA_ROT_IS_MODULO=1), the software limit switches and working-area limitations can be deactivated for individual axes.

A modulo rotary axis with/without working-area limitation can be used.

References:
/FB1/ Function Manual, Basic Functions; Axis Monitoring, Protection Zones (A3)

Mirroring of rotary axes

Mirroring can be implemented for rotary axes by programming `MIRROR(C)` or `AMIRROR(C)`.

Reference point approach

References:
/FB1/ Function Manual, Basic Functions; Reference Point Approach (R1)

Spindles as rotary axes

For notes concerning the use of spindles and rotary axes (C axis operation), please refer to:

References:
/FB1/ Function Manual Basic Functions; Spindles (S1)

13.6 Examples

Fork head, inclined-axis head

Rotary axes are frequently used on 5-axis milling machines to swivel the tool axis or rotate the workpiece. These machines can position the tip of a tool on any point on the workpiece and take up any position on the tool axis. Various milling heads are required, depending on the application. The figure shows a fork head and an inclined-axis head as example rotary-axis arrangements.

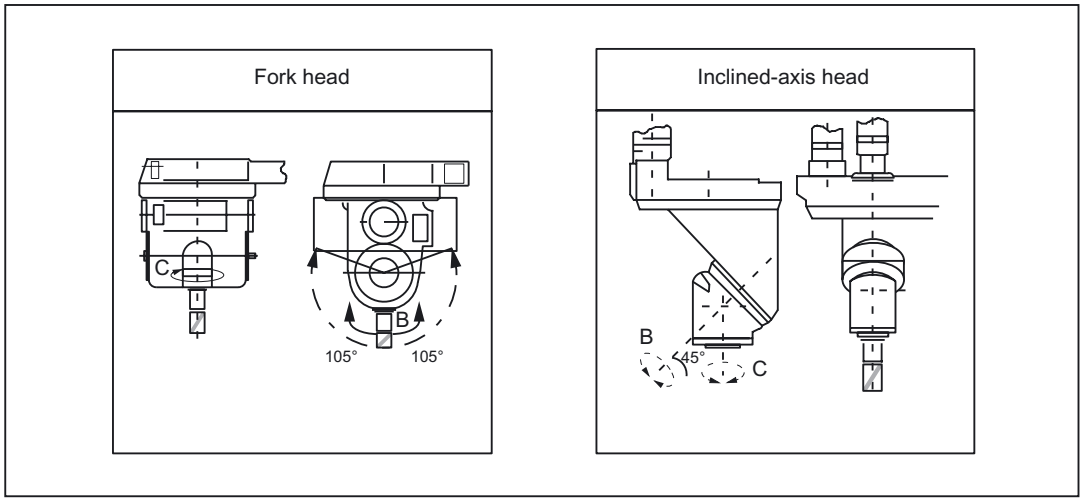


Figure 13-7 Fork head, inclined-axis head

13.7 Data lists

13.7.1 Machine data

13.7.1.1 General machine data

Number	Identifier: $MN_	Description
10210	INT_INCR_PER_DEG	Computational resolution for angular positions

13.7.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Axis is rotary axis
30310	ROT_IS_MODULO	Modulo conversion for rotary axis
30320	DISPLAY_IS_MODULO	Modulo actual-value display
30330	MODULO_RANGE	Modulo-range magnitude
30340	MODULO_RANGE_START	Modulo-range starting position
30455	MISC_FUNCTION_MASK	Axis functions
36100	POS_LIMIT_MINUS	Minus software limit switch
36110	POS_LIMIT_PLUS	Plus software limit switch

13.7.2 Setting data

13.7.2.1 General setting data

Number	Identifier: $SN_	Description
41130	JOG_ROT_AX_SET_VELO	JOG velocity for rotary axes

13.7.2.2 Axis/spindle-specific setting data

Number	Identifier: $SA_	Description
43420	WORKAREA_LIMIT_PLUS	Plus working-area limitation
43430	WORKAREA_LIMIT_MINUS	Minus working-area limitation

13.7.3 Signals

13.7.3.1 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Traversing-range limitation for modulo axis	DB31, ...DBX12.4	DB380x.DBX1000.4

13.7.3.2 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Status of software-limit-switch monitoring for modulo axis	DB31, ...DBX74.4	DB390x.DBX1000.4

S3: Synchronous spindle

14

14.1 Brief description

14.1.1 Function

The "Synchronous spindle" function can be used to couple two spindles with synchronous position or speed. One spindle is defined as leading spindle (LS), the second spindle is then the following spindle (FS).

Speed synchronism:	$n_{FS} = k_{Ü} * n_{LS}$, when $k_{Ü}$ = „1, „2, „3, ...
Position synchronism:	$\varphi_{FS} = \varphi_{LS} + \Delta\varphi$, when 0° x $\Delta\varphi$ t=360°

Possible applications

Rear side machining

One application option is, for example, the reverse side machining in a double-spindle lathe with on-the-fly transfer of the workpiece from the position-synchronous LS to the FS (or vice versa), without having to decelerate down to standstill.

Multi-edge machining (polygonal turning)

The "Synchronous spindle" function provides the basis for multi-edge machining (polygonal turning) through specification of an integer gear ratio $k_{Ü}$ between LS and FS.

Number of FS

The number of FS's that can be operated synchronously to an LS is only restricted by the performance capability of the NC used. In principle, any number of FS can be coupled simultaneously to an LS in arbitrary channels of the NC.

Two pairs of synchronous spindles can be active simultaneously in each NC channel.

Definition

The assignment of FS to LS pair of synchronous spindles can be parameterized channel-specifically via machine data or flexibly defined via part program commands.

Selecting/de-selecting

Part program commands are used to select/deselect the synchronous operation of a pair of synchronous spindles.

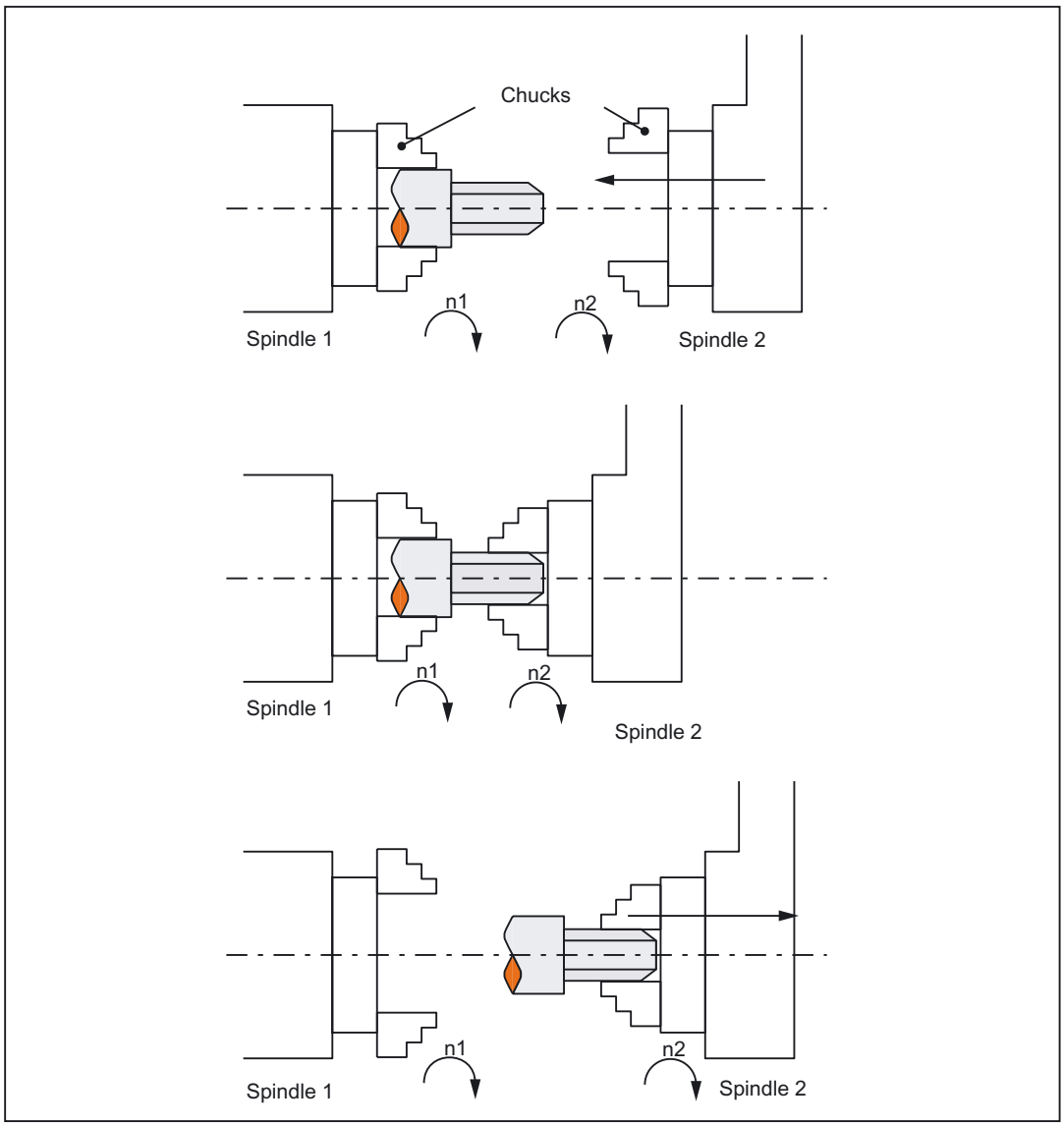


Figure 14-1 Synchronous operation: On-the-fly workpiece transfer from spindle 1 to spindle 2

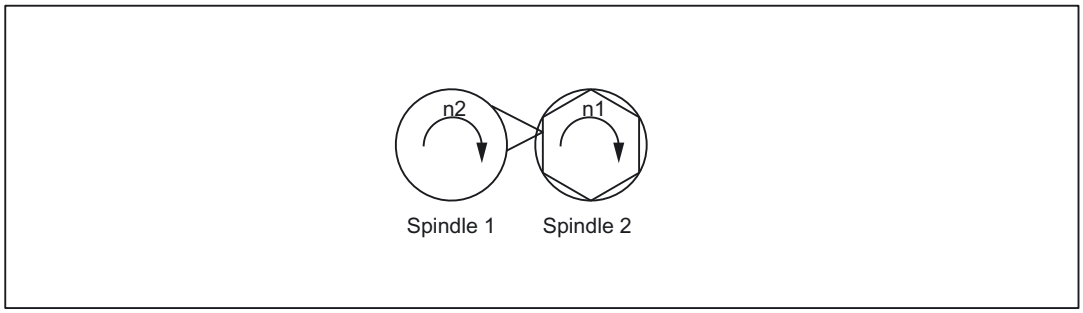


Figure 14-2 Synchronous operation: Polygonal turning

14.1.2 Requirements

The "Synchronous spindle/polygonal turning" option or the corresponding optional generic coupling version is needed to use the function.

Information on the different versions of the generic coupling can be found in:
Reference:
Function Manual Special Functions; Coupled Axes (M3)

14.1.3 Synchronous mode

Description

<axial expression>:	can be: - Axis identifier - Spindle identifier
<Axis identifier>:	C (if spindle has the identifier "C" in axis operation.)
<Spindle identifier>:	Sn, SPI(n) where n = spindle number
<Spindle number>:	1, 2, ... according to the spindle number defined in MD35000 $MA_SPIND_ASSIGN_TO_MACHAX
(FS, LS, offset):	LS = Leading Spindle, FS = Following Spindle,Offset = read programmable offset of following spindle using system variables
$P_COUP_OFFS[Sn]	Programmed position offset of the synchronous spindle

Synchronous spindle pair

Synchronous operation involves a following spindle (FS) and a leading spindle (LS), referred to as the **synchronous spindle pair**. The following spindle imitates the movements of the leading spindle when a coupling is active (synchronous operation) in accordance with the defined functional interrelationship.

Synchronous mode

Synchronous mode (also referred to as “Synchronous spindle operation”) is another spindle operating mode. Before synchronous mode is activated, the following (slave) spindle must have been switched to position control. Synchronous operation is activated for the following spindle when the coupling is activated. As soon as the coupling is deactivated, the following spindle switches to back to open-loop control mode.

As soon as synchronous operation is active for the following spindle, the following interface signal is reported to the PLC:

IS "Synchronous mode" (DB31, ... DBX84.4) = 1.

Number of synchronous spindles

It is possible to couple several following spindles to one leading spindle. The number of following spindles on this leading spindle depends on the respective versions of the appropriate software versions.

Any number of following spindles in any channels of one NCU or a different NCU can be coupled to this leading spindle.

Note that one spindle is always the master and the number of couplings results from the number of axes less the master.

Options in synchronous mode

The following functions are available for synchronous mode:

- FS and LS turn at the same speed

  ($n_{FS} = n_{LS}$; speed ratio $k_{Ü} = 1$)
- Rotation in the same or opposite direction between LS and FS

  (can be defined positively or negatively using speed ratio $k_{Ü}$)
- Following and leading spindles rotate at different speeds

  ($n_{FS} = k_{Ü} \cdot n_{LS}$; speed ratio $k_{Ü} \neq 1$)

  Application: Polygonal turning
- Settable angular position between FS and LS ($\varphi_{FS} = \varphi_{LS} + \Delta\varphi$)

  The spindles run at synchronous speed with a defined angular offset between FS and LS (position synchronous coupling).

  Application: Shaped workpieces
- Activation of synchronous operation between LS and FS can take place when the spindles are in motion or at standstill.
- The full functionality of the open-loop and position control modes is available for the leading spindle.
- When synchronous mode is not active, the FS and LS can be operated in all other spindle modes.
- The speed ratio can also be altered when the spindles are in motion in active synchronous mode.
- With synchronous spindle coupling switched on, the offset of the FS to the LS (overlaid movement) can be altered.

Coupling options

Synchronous spindle couplings can be defined as both

- permanently configured via channel-specific machine data

  (hereinafter referred to as **"permanent coupling configuration"**) as well as
- freely defined using language instructions (COUP...) in the parts program

  (hereafter referred to as **"user defined coupling"**)

The following variants are possible:

1. A fixed configuration for a coupling can be programmed via machine data. In addition, a second coupling can be freely defined via the parts program.
2. No coupling is configured via machine data. In this case, the couplings can be user-defined and parameterized via the parts program.

Separate following spindle interpolator

The separate **following spindle interpolator** allows a number of following spindles from different channels or from another NCU to be coupled as defined by the user to a single leading spindle. The following spindle interpolator is

- `COUPON` or `COUPONC` activated and
- `COUPOF` or `COUPOFS` deactivated

and is always located in the channel in which the `COUPON`, `COUPONC` statement has been programmed for the following spindle. If the following spindle to be activated was previously programmed in another channel, `COUPON`/`COUPONC` initiates an axis replacement and fetches the spindle into its own channel.

Certain synchronous spindle functions can be controlled from the PLC by means of coupling-specific axial VDI interface signals. The latter act exclusively on the slave spindles and do not affect the leading spindle. For more information on this, see "Controlling synchronous spindle coupling via PLC".

Definition of synchronous spindles

Before synchronous operation is activated, the spindles to be coupled (FS, LS) must be defined.

This can be done in two ways depending on the application in question:

1. Permanently configured coupling:

   Machine axes that are to function as the following spindle (FS) and leading spindle (LS) are defined in channel-specific MD 21300 $MC_COUPLE_AXIS_1[n].

   The machine axes programmed as the LS and FS for this coupling configuration cannot be altered by the NC parts program.

   If necessary, the coupling parameters can be modified with the NC parts program.

2. User-defined coupling:

   Couplings can be created and altered in the NC parts program with language instruction "COUPDEF(FS, LS, ...)". If a new coupling relationship is to be defined, it may be necessary to delete an existing user-defined coupling beforehand (with language instruction COUPDEL(FS, LS)).

   The axis identifiers (Sn, SPI(n)) for the following and leading spindles must be programmed with FS and LS for every language instruction COUP..., thus ensuring that the synchronous spindle coupling is unambiguously defined.

   The valid spindle number must then be assigned axis-specific machine data of a machine axis:

   MD35000 $MA_SPIND_ASSIGN_TO_MACHAX.

IS "Following spindle active" (DB31, ... DBX99.1) and IS "Leading spindle active" (DB31, ... DBX99.0) indicate to the PLC for each machine axis whether the axis is active as a leading or following spindle.

The LS can be programmed either via a part program, PLC or also using synchronized actions.

Speed ratio

The speed ratio is programmed with separate numerical values for numerator and denominator (speed ratio parameters). It is therefore possible to specify the speed ratio very exactly, even with rational numbers.

In general:

$k_Ü$ = speed ratio parameter for numerator : Speed ratio parameters for denominator = $Ü_{numerator} : Ü_{denominator}$

The value range of the speed ratio parameter ($Ü_{numerator}$, $Ü_{denominator}$) is virtually unlimited internally in the control.

The speed ratio parameters for the coupling configured via machine data can be defined in channel-specific SD 42300: COUPLE_RATIO_1[n]. In addition, the ratio can be altered with language instruction COUPDEF(FS, LS, $Ü_{numerator}$, $Ü_{denominator}$,...). The values entered in the setting data are not overwritten in this case (default settings).

The ratio for the coupling defined via the NC parts program can only be input with language instruction COUPDEF (...).

The new speed ratio parameters take effect as soon as the COUPDEF instruction has been processed.

For further programming instructions for synchronous spindle couplings, please see "Programming of synchronous spindle couplings" Section.

Coupling characteristics

The following characteristics can be defined for every synchronous spindle coupling:

- **Block change behavior**

  The condition to be fulfilled for a block change can be defined on activation of synchronous operation or on alteration of the ratio or the speed defined angular offset when the coupling is active:

  – Block change takes place immediately
  – Block change in response to "Fine synchronism"
  – Block change in response to "Coarse synchronism"
  – Block change for IPOSTOP (i.e. after setpoint-end synchronism)
  – Check of the synchronism conditions at an arbitrary moment with `WAITC`.

- **Type of coupling** between FS and LS

  The position setpoint or the actual position value of the leading spindle can be used as the reference value for the following spindle. The following coupling types can therefore be selected:

  – Setpoint coupling (DV)

    Use in position controlled operation. The control dynamic response of both spindles should coincide as far as possible. Preferably, the setpoint coupling should be used.

  – Actual value coupling (AV)

    Application if no position control of the LS is possible or with great deviation of the control characteristics between FS and LS. The setpoints for the FS are derived from the actual values of the LS. The quality of synchronism is worse with a varying spindle speed than with the setpoint coupling.

  – Speed coupling (VV)

    Internally, the velocity coupling is a setpoint coupling. The requirements for FS and LS are lower. Position control and measuring systems are not required for FS and LS.

    The position offset between FS and LS is undefined.

The relevant coupling characteristics for the **configured coupling** are selected using machine data, see "Configuration of a synchronous spindle pair via machine data" and for the **user defined coupling** using the language instruction `COUPDEF`, see Section "Preparatory programming instructions".

In addition, coupling characteristics Type of coupling and Block change response can be altered for the permanently configured coupling by means of language instruction `COUPDEF`.

References:

/PGA/ Programming Manual, Job Planning ("Synchronous Spindles").

Change protection for coupling characteristics

The channel-specific MD21340 $MC_COUPLE_IS_WRITE_PROT_1 is used to define whether or not the configured coupling parameters Speed ratio, Type of coupling and Block change response can be altered by the NC parts program:

0: Coupling parameters can be altered by the NC parts program via instruction `COUPDEF`

1: Coupling parameters cannot be altered by the NC parts program. Attempts to make changes will be rejected with an alarm message.

Superimposed motion

In synchronous operation, the synchronous spindle copies the movement of the leading spindle in accordance with the programmed speed ratio.

At the same time, the synchronous spindle can also be traversed with overlay so that the LS and FS can operate at a specific angular position in relation to one another.

The overlaid traversing movement of the FS can be initiated in various ways:

- Programmable position offset of FS for AUTOMATIC and MDA:
  - Language instructions `COUPON` and `SPOS` allow the position reference between FS and LS to be changed while synchronous mode is active, see Section "Selecting synchronous mode of the parts program."
- Manual position offset of FS:
  - In JOG (JOG continuous or JOG incremental) mode

    Superimposition of FS using the handwheel or with plus or minus traversing keys when synchronous operation is active.
  - in AUTOMATIC and MDA modes

    Superimposition of FS with handwheel using DRF offset

As soon as the FS executes the overlaid traversing movement, IS "Overlaid movement" (DB31, ... DBX98.4) is set to the 1 signal.

The overlaid movement is executed optimally in terms of time at the maximum possible FS speed with `COUPON`. With an offset change by means of SPOS, the positioning velocity can be specified with FA[Sn] and manipulated by an override (can be selected through IS "Feedrate override valid for spindle" DB31, ... DBX17.0).

Note

For more information about specifying the position speed with FA[Sn] in:

References:

/FB1/ Function Manual, Basic Functions; Spindles (S1), Section "Spindle modes, positioning operations"

Setpoint correction

The setpoint correction of the system variable $AA_COUP_CORR[Sn] impacts on all subsequent following spindle programming in the same way as a position offset and corresponds to a DRF offset in the MCS.

Example: establish correction value

If a coupling offset of 7° has been programmed using COUPON(....,77) and if a mechanical offset of 81° has come about as a result of closing the workpiece support fixture, a correction value of 4° is calculated:

The system variables return the following values for the following spindle:

$P_COUP_OFFS[S2] ; programmed position offset = 77°

$AA_COUP_OFFS[S2] ; setpoint-end position offset = 77°

$VA_COUP_OFFS[S2] ; actual value-end position offset approx. 77°

$AA_COUP_CORR[S2] ; correction value = 4°

14.1.4 Prerequisites for synchronous mode

Conditions on selection of synchronous mode

The following conditions must be fulfilled before the synchronous spindle coupling is activated or else alarm messages will be generated.

- The synchronous spindle coupling must have been defined beforehand (either permanently configured via machine data or according to user definition via parts program using `COUPDEF`).
- The spindles to be coupled must be defined in the NC channel in which the coupling is activated.

  Channel-specific MD20070 $MC_AXCONF_MACHAX_USED

  Axis-specific MD35000 $MA_SPIND_ASSIGN_TO_MACHAX
- The following spindle must be assigned to the NC channel in which the coupling is activated.

  Default setting with axis-specific MD30550 AXCONF_ASSIGN_MASTER_CHAN
- The following applies to setpoint and actual value couplings (DV, AV):

  FS and LS must at least have a position measuring system for recording positions and position controls must be started up.

  Note

  When position control is activated, the maximum setpoint speed of the LS is automatically limited to 90% (control reserve) of the maximum speed. The limitation is signaled via IS "Setpoint speed limited" (DB31, ... DBX83.1).

References:

/FB1/ Function Manual Basic Functions; Spindles (S1)

- The following applies to setpoint couplings (DV):

  To ensure more accurate synchronization characteristics, the LS should be in position control mode (language instruction SPCON) before the coupling is activated.

- Before selecting the synchronous mode, the gear stage necessary for FS and LS must be selected. In synchronous mode, gear stage changeover and therefore oscillation mode are not possible for FS and LS. Upon request, an alarm message is generated.

Cross-channel coupling

The LS can be located in any channel.

- The LS can be exchanged between channels by means of "Axis exchange".
- When several following spindles are coupled to one leading spindle, the dynamic response of the coupling is determined by the weakest response as a function of the coupling factor. The acceleration rate and maximum speed are reduced for the leading spindle to such a degree that none of the coupled following spindles can be overloaded.
- The following spindle is always located in the channel in which the coupling has been activated using COUPON or COUPONC.

14.1.5 Selecting the parts program's synchronous mode

Activate coupling COUPON, COUPONC

Language instruction COUPON activates the coupling in the parts program between the programmed spindles with the last valid parameters and thus also activates synchronous mode. This coupling may be a fixed configuration or user-defined. The leading spindle and/or following spindle may be at standstill or in motion at the instant of activation.

Certain conditions must be fulfilled before synchronous operation can be activated, see Section "Prerequisites for synchronous mode".

The COUPONC statement adopts the previous programmed direction of spindle rotation and spindle speed for the following and leading spindle in the parts program. It is not possible to specify an angular offset.

COUPON activation variants

Two different methods can be selected to activate synchronous mode:

1. Fastest possible activation of coupling with **any angular reference** between leading and following spindles.

   COUPON(FS, LS)

2. Activation of coupling with a **defined angular offset** POS_{FS} between leading and following spindle. With this method, the angular offset must be programmed on selection.

   COUPON(FS, LS, POS_{FS})

Note

If the LS and/or FS is in axis mode before switching on the synchronous coupling, the axis mode is left and spindle mode is activated with use of the spindle identifier.

If the spindle is switched on with use of the axis identifier, no changeover takes place.

Block change behavior

Before synchronous operation is selected, it must be determined under what conditions the block change must occur when synchronous mode is activated, see Section "Preparatory programming instructions".

Determining current coupling status

It is possible to determine the current coupling status for the specified axis/spindle in the NC parts program by means of axial system variable $AA_COUP_ACT[<axial expression>], see Section "Axial system variables for synchronous spindles". As soon as the synchronous spindle coupling is active for the following spindle, bit 2 must be "1" when read.

Change defined angular offset

Language instructions `COUPON` and `SPOS` allow the defined angular offset to be changed while synchronous mode is active. The following spindle is positioned as an overlaid movement at the angular offset programmed with POS_{FS}. During this time, the IS "Superimposed movement" (DB31, ... DBX98.4) is set.

Angular offset POS_{FS}

The defined angular offset POS_{FS} must be specified as an absolute position referred to the zero degrees position of the leading spindle in a positive direction of rotation.

The "0° position" of a position-controlled spindle is calculated as follows:

- from the zero mark or Bero signal of the measurement system and
- from the reference values saved using axis-specific machine data:

  MD34100 $MA_REFP_SET_POS, reference point value,

  of no significance with interval-coded systems.

  MD34080 $MA_REFP_MOVE_DIST reference point distance/target point

  with interval-coded systems,

  MD34090 $MA_REFP_MOVE_DIST_CORR reference point offset/absolute offset with interval coding.

Range of POS_{FS}: 0 ... 359,999°.

References:

/FB1/ Function Manual, Basic Functions; Reference Point Approach (R1)

Read current angular offset

Using axial system variables, it is possible to read the current position offset between the FS and LS in the NC parts program. The following two position offsets exist:

- Current position offset of setpoint between FS and LS

  $AA_COUP_OFFS [<axis identifier for FS>]

- Current position offset of actual value between FS and LS

  $VA_COUP_OFFS [<axis identifier for FS>]

(Explanation of <axis identifier>, see section "Synchronous operation")

Activation after power ON

Synchronous mode can also be activated with non-referenced/synchronized FS or LS (IS "Referenced/synchronized 1 or 2" DB31, ... DBX60.4 or DBX60.5 = 0). In this case, a warning message is displayed.

Example:

LS and FS are already coupled in a friction lock via a workpiece after power ON.

14.1.6 Deselecting the parts program's synchronous mode

Deactivate coupling COUPOF, COUPOFS

Synchronous mode between the specified spindles is canceled by the parts program instruction COUPOF. Three variants are possible.

If synchronous mode is canceled between the specified spindles using COUPOF, then it is irrelevant whether this coupling is permanently configured or user defined. The leading and following spindles can be at standstill or in motion when synchronous operation is deactivated.

On switching off the synchronous mode with COUPOF, the following spindle is put into **control mode**. The originally programmed S-word is no longer valid for the FS, the following spindle can be operated like any other normal spindle.

When the coupling is deactivated with COUPOF, a block preprocessing stop STOPRE is generally initiated internally in the control.

The COUPOFS instruction can be used to deactivate a coupling either as quickly as possible with a stop and no position details or with a stop at the programmed position.

COUPOF variants

Three different methods can be used to deselect synchronous mode with COUPOF:

1. Deactivation of coupling as quickly as possible

   The block change is enabled immediately.

   COUPOF(FS, LS)

2. A coupling is not deselected until the following spindle has crossed the programmed deactivation position POS_{FS}.

   The block change is then approved.

   COUPOF(FS, LS, POS_{FS})

3. A coupling is not deselected until the following spindle and leading spindle has crossed the programmed deactivation positions POS_{FS} and POS_{LS}.

   The block change is then approved.

   COUPOF(FS, LS, POS_{FS}, POS_{LS})

Note

If the LS and/or FS is in axis mode before switching off the synchronous coupling, the axis mode is left with use of the spindle identifier and the speed control mode is activated.

If the spindle is switched off with use of the axis identifier, no changeover takes place. Before shutdown, the LS must be in the setpoint-side standstill.

POS_{FS}, POS_{LS}

Deactivation positions POS_{FS} and POS_{LS} match the actual positions of FS and LS respectively referred to the defined reference point value.

Range of POS_{FS}, POS_{LS}: 0 ... 359,999°.

References:

/FB1/ Function Manual, Basic Functions; Reference Point Approach (R1)

COUPOF during the motion

If synchronous mode is deselected while the spindles are in motion with `COUPOF`, the following spindle continues to rotate at the current speed (n_{FS}). The current speed can be read with system variable $AA_S in the NC parts program.

The following spindle can then be stopped from the parts program with M05, `SPOS`, `SPOSA` or from the PLC with the appropriate interface signal.

COUPOFS with stop of following spindle

Another deactivation method for a synchronous spindle coupling, i.e. by stopping the following spindle, has been added:

- Deactivating a coupling as quickly as possible and stop without specifying the position.

  The block change is then approved.

  COUPOFS(FS, LS)

- Deactivating the coupling with stop of following spindle at the programmed position. The block change is then approved.

Condition:

COUPOFS(FS, LS) and COUPOFS(FS, LS, POS_{FS}) have no meaning if a coupling was active.

14.1.7 Controlling synchronous spindle coupling via PLC

Controlling following spindle via PLC

Using the coupling-specific, axial VDI interface signals, it is possible to control synchronization motions for the following spindle from the PLC program. This offers the option of utilizing the PLC to disable, suppress or restore a synchronization motion for the following spindle specified by offset programming.

These signals have no effect on the leading spindle. The following coupling-specific VDI signal (PLC → NCK) is available:

IS **"Disable synchronization" (DB31, ... DBX31.5)**

"Disable synchronization"

The synchronization motion for the following spindle is suppressed using the axial signal IS "Disable synchronization" (DB31, ... DBX31.5).

When the main run advances to a block containing parts program statement COUPON (FS, LS, offset), the following interface signal is evaluated for the following spindle:

IS "Disable synchronization" (DB31, ... DBX31.5).

- For IS "Disable synchronization" (DB31, ... DBX31.5) = 0, the position offset is traversed through as before.
- For IS "Disable synchronization" (DB31, ... DBX31.5) = 1, only the continuous velocity synchronism is established. The following spindle does not execute any additional movement.

  The coupling then responds analogously to a programmed COUPON(<FS>,<LS>).

Special features

For the IS "Disable synchronization" (DB31, ... DBX31.5) offset motion of the following spindle cannot be controlled that was generated as follows:

- SPOS, POS
- Synchronized actions
- FC18 (for 840D sl)
- JOG

These functions are controlled by VDI signal IS "Feedrate stop/Spindle stop" (DB31, ... DBX4.3).

Synchronized state reached

Whenever a state of synchronism has been reached, the following two VDI signals are set regardless of whether synchronization has been disabled or not:

IS "Synchronism coarse" (DB31, ... DBX98.1) and

IS "Synchronism fine" (DB31, ... DBX98.0)

Further block changes after COUPON are not prevented by suppression of synchronization.

Example

Block change behavior after COUPON

```
                                        ; IS "Disable synchronization"
                                        ; set (DB31, ... DBX31.5) = 1 for S2
N51 SPOS=10 SPOS[2]=10                  ; Positions correspond to an offset
                                          of 0°
N52 COUPDEF(S2,S1,1,1,"FINE","DV")
N53 COUPON(S2,S1,77)                    ; Actual offset of 0 degrees is retained
                                        ; no following spindle movement,
                                        ; VDI signals
```

```
                                                    ; IS "Synchronism coarse"
                                                    ; (DB31, ... DBX98.1) and
                                                    ; IS "Synchronism fine"
                                                    ; (DB31, ... DBX98.0)
                                                    ; are set and the block change
                                                    ; is enabled.
N54 M0
N57 COUPOF(S2,S1)
N99 M30
```

Reset and recovery

Resetting the IS "Disable synchronization" (DB31, ... DBX31.5) has no effect on the following spindle offset. If the offset motion of the following spindle has been suppressed by the VDI interface signal, then the offset is not automatically applied when the VDI signal is reset.

Synchronization is recovered as follows:

- By repeating the part program operation `COUPON` (FS, LS, offset) with IS "Disable synchronization" (DB31, ... DBX31.5) = 0.

  `COUPON` (FS, LS, offset) can be written e.g. in an ASUB.
- By setting the IS "Resynchronize" (DB31, ... DBX31.4) = 1

Read offset

The following system variables can be used to read three different position offset values of the following spindle from the parts program and synchronized actions. The variable $P_COUP_OFFS[Sn] is only available in the parts program.

Description	NCK variable
Programmed position offset of the synchronous spindle	$P_COUP_OFFS[Sn]
Position offset of synchronous spindle, setpoint end	$AA_COUP_OFFS[Sn]
Position offset of synchronous spindle, actual value end	$VA_COUP_OFFS[Sn]

"Feedrate stop/spindle stop"

By configuring bit 4 in MD30455 MISC_FUNCTION_MASK, the behavior of the axial IS **"Feedrate stop/Spindle stop" (DB31, ... DBX4.3)** is defined for the following spindle.

Bit 4 = 0 compatibility method:

Canceling feed enable for the following spindle decelerates the coupling assembly.

Bit 4 = 1:

Feedrate enable refers only to the interpolation component (SPOS),..) and does not affect the coupling.

Note

Other configuration options for axis functions using MD30455 $MA_MISC_FUNCTION_MASK:

References:

/FB1/ Function Manual, Basic Functions; Round Axes (R2), Section "Programming Round Axes

14.1.8 Monitoring of synchronous operation

Fine/coarse synchronism

In addition to conventional spindle monitoring operations, synchronous operation between the FS and LS is also monitored in synchronous mode.

For this, IS "Synchronism fine" (DB31, ... DBX98.0) or IS "Synchronism coarse" (DB31, ... DBX98.1) is transmitted to the PLC to indicate whether the current position (AV, DV) or actual speed (VV) of the following spindle is within the specified tolerance window.

When the coupling is switched on, the signals "Coarse synchronism" and "Fine synchronism" are updated when setpoint synchronism is reached.

The size of the tolerance windows is set with machine data of the FS. Reaching of the synchronism is influenced by the following factors:

- AV, DV: Position variance between FS and LS
- VV: Difference in speed between FS and LS

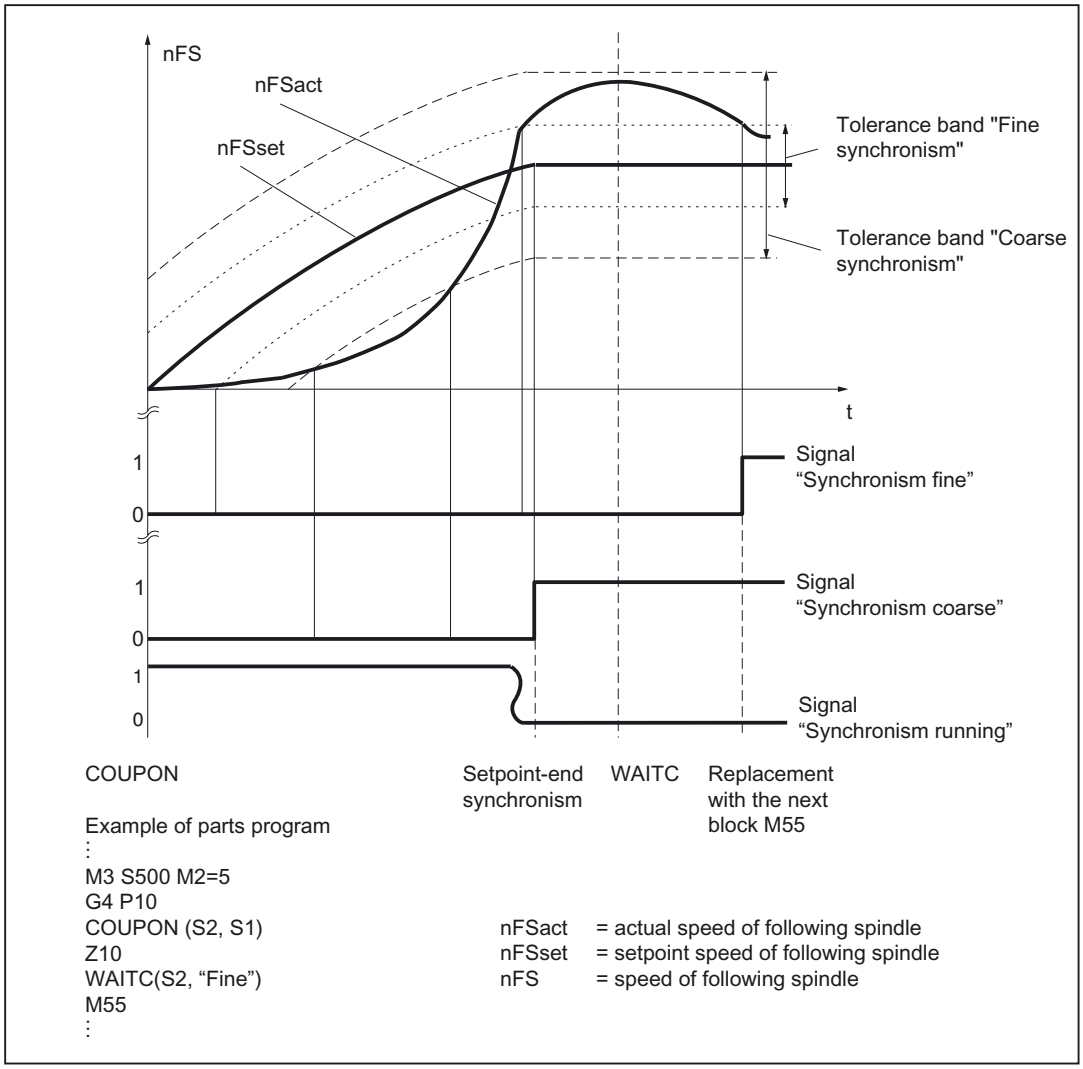


Figure 14-3 Synchronism monitoring with `COUPON` and synchronism test mark `WAITC` with synchronization on a turning leading spindle

Threshold values

The relevant position or velocity tolerance range for the following spindle in relation to the leading spindle must be specified in degrees or 1 rev/min.

- Threshold value for "Coarse synchronism"

  Axis-specific MD37200: AV, DV: COUPLE_POS_TOL_COARSE

  MD37220: VV: COUPLE_VELO_TOL_COARSE

- Threshold value for "Fine synchronism"

  Axis-specific MD37210: AV,DV: COUPLE_POS_TOL_FINE

  MD37230: VV: COUPLE_VELO_TOL_FINE

Speed/acceleration limits

In synchronous mode, the speed and acceleration limit values of the leading spindle are adjusted internally in the control in such a way that the following spindle can imitate its movement, allowing for the currently selected gear stage and effective speed ratio, without violating its own limit values.

For example, the LS is automatically decelerated to prevent the FS from exceeding the maximum speed in order to maintain synchronism between the spindles.

14.2 Programming of synchronous spindle couplings

Table 14- 1 Overview

Programmed coupling	Configured coupling(s)	Note
Defining a coupling: **COUPDEF(FS, ...)**	Modification of configured data: **COUPDEF(FS, ...)**	Setting the coupling parameters
Activation of a coupling: **COUPON(FS, LS, POS$_{FS}$)** Activate and transfer a movement for coupling difference in speed: **COUPONC(FS, LS)**		Switching on and switching off
Deactivation of a coupling: **COUPOF(FS, LS, POS$_{FS}$, POS$_{LS}$)** with stop of following spindle: **COUPOFS(FS, LS, POS$_{FS}$)**		
Deleting the coupling data: **COUPDEL (FS, LS)**	Reactivating the configured data: **COUPRES (FS, LS)**	Clearing up, restoring

Reduced specification without the main spindle

Without specifying a leading axis, the the following language instructions are permitted:

COUPOF(FS), COUPOFS(FS), COUPDEL(FS), COUPRES(FS).

Note

The FS and LS must be programmed for each COUPDEF, COUPON and COUPONC instruction so that alarm messages are not triggered.

References:

/PGA/, Programming Manual Job Planning, "Synchronous Spindles"

14.2.1 Preparatory programming instructions

Programmable couplings

The number of couplings can be programmed as often as desired depending on the axes available. This number results from the number of axes/spindles less one for the master. Furthermore, one coupling can also be configured via machine data as in earlier SW versions.

Permanently configured coupling

The coupling characteristics and speed ratio for a permanently configured synchronous spindle coupling can be altered by the NC parts program provided that they are not write-protected. The machine axes for LS and FS cannot be changed.

Define new couplings

Language instruction "COUPDEF" can be used to create new synchronous spindle couplings (user-defined) and to modify the parameters for existing couplings.

When the coupling parameters are fully specified, the following applies:

COUPDEF (FS, LS, $Ü_{numerator}$, $Ü_{denominator}$, block change response, coupling type)

The spindle coupling is unambiguously defined with FS and LS

The other coupling parameters must only be programmed when they need to be changed. The last valid status remains applicable for non-specified parameters.

The individual coupling parameters are explained below:

- **FS, LS**: Spindle identifiers for following and leading spindles

  e.g.: S1, SPI(1), S2, SPI(2)

  The valid spindle number must be assigned in the axis-specific MD35000 $MA_SPIND_ASSIGN_TO_MACHAX of a machine axis.

- $Ü_{numerator}$, $Ü_{denominator}$: Speed ratio parameters for numerator and denominator

  The speed ratio is specified in the form of numeric values for numerator and denominator, see Section "Synchronous operation".

  The numerator must always be programmed. If no denominator is specified, then its value is always assumed to be "1.0".

- **Block change behavior**

  This parameter allows you to select when the block change should take place when synchronous operation is selected:

NOC: Block change is enabled immediately

FINE: Block change under "Fine synchronism"

COARSE: Block change in response to "Coarse synchronism"

IPOSTOP: Block change for IPOSTOP (i.e. after setpoint-end synchronism)

The block change response is specified as a character string (i.e. with quotation marks).

The block change response can be specified simply by writing the letters in bold print. The remaining letters can be entered to improve legibility of the parts program but they are not otherwise significant.

If no block change response is specified, then the currently selected response continues to apply.

With the programmable synchronism test markers **WAITC**, the replacement with new blocks is delayed until the parameterized synchronism is reached.

- **Coupling type**

DV (Desired Values): Setpoint coupling between FS and LS

AV (Actual Values): Act.-val. coupl. between FS and LS

VV (Velocity Values): Speed coupling between FS and LS

If no coupling type is specified, then the currently selected type continues to apply.

Note

The coupling type may only be changed when synchronous operation is deactivated!

Examples

COUPDEF (SPI(2), SPI(1), 1.0, 1.0, "FINE", "DV")

COUPDEF (S2, S1, 1.0, 4.0)

COUPDEF (S2, SPI(1), 1.0)

Default settings

The following default settings apply to user-defined couplings:

- $Ü_{Numerator}$=1.0
- $Ü_{Denominator}$ =1.0
- Block change response = **IPOSTOP** (block change enabled with setpoint synchronism)
- Type of coupling = **DV** (setpoint coupling)

Delete couplings

Language instruction "COUPDEL" is used to delete user-defined couplings.

COUPDEL (FS, LS)

Note

COUPDEL impacts on an active coupling, deactivates it and deletes the coupling data. Alarm 16797 is therefore meaningless.

The following spindle adopts the last speed. This corresponds to the behavior associated with COUPOF(FS, LS).

Activate original coupling parameters

Language instruction "COUPRES" can be used to re-activate the configured coupling parameters.

COUPRES (FS, LS)

The parameters modified using COUPDEF (including the speed ratio) are subsequently deleted.

Language instruction "COUPRES" activates the parameters stored in the machine and setting data (configured coupling) and activates the default settings (user-defined coupling).

Programmable block change

It is possible to mark a point in the NC program using the "WAITC" language instruction. The system waits at this point for fulfillment of the synchronism conditions for the specified FS and delays changes to new blocks until the specified state of synchronism is reached (see Fig.).

WAITC (FS)

Advantage: The time between activating the synchronous coupling and reaching synchronism can be used in a meaningful way, technologically speaking.

Note

Basically, it is always possible to write WAITC. If the spindle indicated is not active as FS, the instruction for this spindle is without effect.

If no synchronism condition is indicated, the check is always performed for the synchronism condition programmed/configured on the respective coupling, at least for the setpoint synchronism.

Examples:

```
WAITC(S2),
:
WAITC(S2, "Fine"),
:
WAITC(S2, ,S4, "Fine")
```

Stop and block change

If "Stop" has been activated for the cancellation period of the axis enables for the leading or following spindle, then the **last** setpoint positions with the setting of the axis enables from the servo drive are approached again.

Program instructions COUPON and WAITC can influence the block change behavior. The block change criterion is defined using COUPDEF or via the MD21320 $MC_COUPLE_BLOCK_CHANGE_CTRL_1.

14.2.2 Programming instructions for activating and deactivating the coupling

Activate synchronous mode

Language instruction COUPON is used to activate couplings and synchronous mode.

Two methods by which synchronous operation can be activated are available:

1. **COUPON(FS, LS)**

   Fastest possible activation of synchronous operation with any angular reference between the leading and following spindles.

2. **COUPON(FS, LS, POS$_{FS}$)**

   Activation of synchronous operation with a defined angular offset POS$_{FS}$ between the leading and following spindles. This offset is referred to the zero degrees position of the leading spindle in a positive direction of rotation. The block change is enabled according to the defined setting. Range of POS$_{FS}$: 0 ... 359.999 degrees.

3. **COUPONC(FS, LS)**

   When activating with `COUPONC`, the previous programming of M3 S... or M4 S... is adopted. A difference in speed is transferred immediately. An offset position cannot be programmed.

By programming `COUPON`(FS, LS, POS$_{FS}$) or SPOS when synchronous operation is already active, the angular offset between LS and FS can be changed.

Deactivate synchronous mode

Three different methods can be selected to deactivate synchronous mode:

1. **COUPOF(FS, LS)**

   Fastest possible deactivation of synchronous operation. The block change is enabled immediately.

2. **COUPOF(FS, LS, POS$_{FS}$)**

   Deselection of synchronous operation after deactivation position POS$_{FS}$ has been crossed. Block change is not enabled until this position has been crossed.

3. **COUPOF(FS, LS, POS$_{FS}$, POS$_{LS}$)**

   Deselection of synchronous operation after the two deactivation positions POS$_{FS}$ and POS$_{LS}$ have been crossed. Block change is not enabled until **both** programmed positions have been crossed.

   Range of POS$_{FS}$, POS$_{LS}$: 0 ... 359,999°.

If continuous path control (G64) is programmed, a non-modal stop is generated internally in the control.

Examples:

```
COUPDEF (S2, S1, 1.0, 1.0, "FINE, "DV")
:
COUPON (S2, S1, 150)
:
COUPOF (S2, S1, 0)
:
COUPDEL (S2, S1)
```

1. **COUPOFS(FS, LS)**

   Deactivating a coupling with stop of following spindle. Block change performed as quickly as possible with immediate block change)

2. **COUPOFS(FS, LS, POS$_{FS}$)**

   After the programmed deactivation position that refers to the machine coordinate system has been crossed, the block change is not enabled until the deactivation positions POSFS have been crossed.

   Value range 0 ... 359,999°.

14.2.3 Axial system variables for synchronous spindle

Determining current coupling status

The current coupling status of the following spindle can be read in the NC parts program with the following axial system variable:

$AA_COUP_ACT[<axial expression>]

For explanation of <axial expression> refer to " Synchronous mode (Page 875) ".

Example:

$AA_COUP_ACT[S2]

The value read has the following significance for the following spindle:

Byte = 0:	No coupling active
Bit 2 = 1:	Synchronous spindle coupling active
Bit 2 = 0:	Synchronized spindle coupling is not active

Read current angular offset

The current position offset between the FS and LS can be read in the NC part program by means of the following axial system variables:

- Setpoint-based position offset between FS and LS:

  $AA_COUP_OFFS[<axial expression>]

- Actual-value-based position offset between FS and LS:

  $VA_COUP_OFFS[<axial expression>]

Example:

$AA_COUP_OFFS[S2]

If an angular offset is programmed with `COUPON`, this coincides with the value read after reading the setpoint synchronization.

Reading the programmed angular offset

The position offset last programmed between the FS and LS can be read in the NC part program by means of the following axial system variables:

$P_COUP_OFFS[<axial expression>]

Note

After cancellation of the servo enable signal when synchronous operation and follow-up mode are active, the position offset applied when the controller is enabled again is different to the originally programmed value.

$P_COUP_OFFS only returns the value originally programmed. $AA_COUP_OFFS and $VA_COUP_OFFS return the current value. The programmed offset can be recreated with NST DB31, ... DBX31.4 (re-synchronization).

14.2.4 Automatic selection and deselection of position control

Behavior in speed control mode

In DV coupling mode, program instructions `COUPON`, `COUPONC` and `COUPOF`, `COUPOFS` are used to activate and/or deactivate position control for the leading spindle as required. If there are several following spindles on the leading spindle, then in speed-controlled mode, the **first** DV **activates** coupling position control for the leading spindle and the **last** DV coupling **deactivates** coupling position control for the leading spindle if SPCON is not programmed.

The leading spindle does not need to be located in the same channel as the following spindle.

Automatic selection with COUPON and COUPONC

Depending on the coupling type, the effect of `COUPON` and `COUPONC` on the position control for synchronous operation is as follows:

Coupling type	DV	AV	VV
Following spindle FS	Position control ON	Position control ON	No action
Leading spindle LS	Position control On [1]	No action	No action

[1] The position control is activated by a `COUPON` and `COUPONC` instruction if **at least one** following spindle has been coupled to it with coupling type DV.

Automatic deselection with COUPOF and COUPOFS

Depending on the coupling type, the effect of COUPOF and COUPOFS on the position control is as follows:

Coupling type	DV	AV	VV
Following spindle FS	Position control OFF [2]	Position control OFF [2]	No action [2]
Leading spindle LS	Position control OFF [3]	No action	No action

[2] COUPOF and COUPOFS without position specification

Speed control mode is activated for the following spindle. Positioning mode is activated with COUPFS with a stop position. Position control is **not deactivated** if the following spindle was located in position-controlled spindle mode using SPCON or COUPFS was programmed with position.

[3] With COUPOF position control is **deactivated** if there are no more couplings of the DV coupling type for this leading spindle. Position control **is retained** if the leading spindle is in positioning mode or axial mode **or** was in position-controlled spindle mode using SPCON.

14.3 Configuration of a synchronous spindle pair via machine data

Coupling parameters

One synchronous spindle coupling per NC channel can be configured permanently via channel-specific machine data.

It is then necessary to define the machine axes (spindles) which are to be coupled and what characteristics this coupling should have.

The following parameters can be configured as fixed settings for the synchronous spindle coupling:

- **Synchronous spindle pair** (channel-specific MD21300 $MC_COUPLE_AXIS_1[n])

  This machine data defines the two machine axes which are to form the synchronous spindle pair (following spindle (n=0), leading spindle (n=1)).

  An entry of 0 as the setting for the axis number means that no coupling is permanently configured via the machine data. The machine data for the coupling characteristics are then irrelevant.

  The machine axis numbers for the LS and FS can not be changed by the NC parts program for a configured coupling configuration.

- **Speed ratio**

  This is entered via setting data using two speed ratio parameters (channel-spec. SD42300 $SC_COUPLE_RATIO_1[n]) in the form of a numerator and a denominator.

  $k_Ü$ = speed ratio parameter numerator Speed ratio parameters for denominator = $SC_COUPLE_RATIO[0] : $SC_COUPLE_RATIO[1]

  Provided it is not write-protected, the speed ratio can be changed by the NC parts program with language instruction `COUPDEF`.

- **Block change behavior**

  (channel-specific MD21320 $MC_COUPLE_BLOCK_CHANGE_CTRL_1)

  One of the following options can be selected as the condition for a block change:

  0: Block change takes place immediately

  1: Block change in response to "Fine synchronism"

  2: Block change in response to "Coarse synchronism"

  3: Block change for `IPOSTOP` (i.e. after setpoint-end synchronism)

- **Type of coupling** between FS and LS

  (channel-specific MD21310 $MC_COUPLING_MODE_1)

  0: Actual value coupling (AV)

  1: Setpoint coupling (DV)

  2: Speed coupling (VV)

- **Aborting the coupling with NC start**:

  channel-specific MD21330 $MC_COUPLE_RESET_MODE_1

- **Write-protection for coupling parameters:**

  (channel-specific MD21340 $MC_COUPLE_IS_WRITE_PROT_1)

  It can be defined in this machine data whether or not the configured coupling parameters Speed ratio, Type of coupling and Block change response may be influenced by the NC parts program.

  0: Coupling parameters can be changed by the NC parts program

  1: Coupling parameters cannot be changed by the NC parts program. Attempts to make changes are rejected with an alarm message.

14.3.1 Configuration of the behavior with NC start

The response to NC machining program start is defined by the channel-specific machine data.

Table 14- 2 Synchronous coupling behavior with NC start

	Configured coupling	Programmed coupling *
	MD COUPLE_RESET_MODE	MD START_MODE_MASK
Coupling is maintained	Bit 0 = 0	Bit 10 = 0
Deselect coupling	Bit 0 = 1	Bit 10 = 1
Activate configured data	Bit 5 = 1	-
Activate coupling	Bit 9 = 1	-

*see Section "Configuration of a synchronous spindle pair via machine data"

14.3.2 Configuration of the behavior with Reset

The following behavior can be set with the channel-specific machine data upon reset and end of NC machining program:

Table 14- 3 Synchronous coupling behavior with end of NC machining program and after reset

	Configured coupling	Programmed coupling *
Coupling is maintained	MD COUPLE_RESET_MODE Bit 1 = 0	MD RESET_MODE_MASK Bit 10 = 1
Deselect coupling	MD COUPLE_RESET_MODE Bit 1 = 1 MD RESET_MODE_MASK Bit 0 = 1 (producing a block with RESET)	MD RESET_MODE_MASK Bit 10 = 0 Bit 0 = 1
Activate configured data	MD COUPLE_RESET_MODE Bit 6 = 1 MD RESET_MODE_MASK Bit 0 = 1	-

*see Section "Configuration of a synchronous spindle pair via machine data"

14.4 Special features of synchronous mode

14.4.1 Special features of synchronous mode in general

Control dynamics

When using the setpoint coupling, the position control parameters of FS and LS (e.g. K_V factor) should be matched with one another. If necessary, different parameter blocks should be activated for speed control and synchronized mode. As a variance from position control, feedforward control and parameter block, the control parameters of the following spindle can also be set as they would be for an uncoupled scenario using MD30455 $MA_MISC_FUNCTION_MASK, see Section "Special points regarding start-up of a synchronous spindle coupling".

Feedforward control

Due to the improved control system dynamic response it provides, feedforward control for the following and leading spindles in synchronous mode is **always active.**

It can, however, be deselected for FS and LS with axis-specific MD32620 $MA_FFW_MODE. If MD32620 $MA_FFW_MODE is set to zero, there are function limitations. Position control can no longer be activated in motion with SPCON. SPOS, M19 or SPOSA are therefore not possible. The NC parts program cannot deactivate the feedforward control for LS and FS with FFWOF.

The feedforward control mode (speed or torque feedforward control) is defined in axis-specific MD32620 $MA_FFW_MODE.

References:

/FB2/ Function Manual, Extended Functions, Compensations (K3)

Speed and acceleration limits

The speed and acceleration limits of the spindles operating in synchronous mode are determined by the "weakest" spindle in the synchronous spindle pair. The current gear stages, the programmed acceleration and, for the leading spindle, the effective position control status (On/Off) are taken into account for this purpose.

The maximum speed of the leading spindle is calculated internally in the control taking into account the speed ratio and the spindle limitations of the following spindle.

Multiple couplings

If the system detects that a coupling is already active for an FS and LS when synchronous mode is activated, then the activation process is ignored and an alarm message is generated.

Example of multiple couplings:

- A spindle is acting as the FS for several LS.
- Coupling cascade

Number of configurable spindles per channel:

- Every axis in the channel can be configured as a spindle. The number of axes per channel depends on the control version.

Cross-channel setpoint coupling and optional number of following spindles in optional channels of an NCU:

- Cross-channel synchronous spindle couplings can be implemented with no additional restrictions for DV, AV, and VV.
- Any number of following spindles, corresponding to the number of all spindles minus one spindle for the master, in any channels on an NCU can be coupled to one leading spindle. The real time CPU time needed may possibly also impact on the number of following spindles.

Start synchronous mode using ASUB

ASUBs (activation of asynchronous subprograms) processed by the PLC can be used to activate or terminate synchronous mode at any chosen time in the AUTOMATIC or MDA modes.

References:

/FB1/ Function Manual, Basic Functions, Mode Group, Channel, Program Operation, Reset Response (K1)

Response to alarms

If alarms occur (e.g. servo alarms) **during the synchronous operation**, with which the servo enable is cancelled internally in the control and follow-up is active, the subsequent response is as if the IS "Servo enable" (DB31, ... DBX2.1) had been removed by the PLC (and IS "Follow-up mode" (DB31, ... DBX1.4) is set), see Section "Influence on synchronous operation via PLC interface".

IS "Re-synchronization" (DB31, ... DBX31.4) can be used to reproduce the programmed offset, see Section "Reproduce synchronism of following spindle".

14.4.2 Restore synchronism of following spindle

Causes for a positional offset

When the coupling is reactivated after the drive enable signals have been canceled, a positional offset can occur between the leading and following spindles if follow-up mode is activated. A positional offset can be caused by:

- A part has been clamped or both spindles have been turned manually (machine area is open, drives are disconnected from supply).
- After the spindle enable signals are canceled, the two spindles coast to standstill at different speeds if they are not mechanically coupled.
- A drive alarm occurs (internal follow-up mode):

  DB31, ... DBX61.3 (follow-up mode active) = 1

  When the alarm is cleared, the NC must not trigger any synchronization motion.
- A synchronization was not executed due to a synchronization lock of the following spindle:

  DB31, ... DBX29.5 (Disable synchronization)

Basic procedure

If the following and leading spindles have fallen out of synchronism, or failed to synchronize at all, synchronism can be restored between them by the following measures:

1. Set the axis enable signals and cancel synchronization disable signal if this has been set.
2. Start following spindle resynchronization with the NC/PLC interface signal:

   DB31, ... DBX31.4 (resynchronization)

   Only after the re-synchronization process is complete can the setpoint-end synchronism be fully restored.
3. Wait until the coupled spindles have synchronized.

Enable resynchronization

Setting the enabling signals closes the coupling at the current actual positions. The two following NC/PLC interface signals are set:

DB31, ... DBX98.1 (coarse synchronism)

DB31, ... DBX98.0 (fine synchronism)

The following **preconditions** must be fulfilled for resynchronization to work:

- The axis enabling signal must be set for the following spindle.
- The PLC must not set any synchronization disables for the following spindle:

  DB31, ... DBX31.5 (Disable synchronization)

Resynchronize following spindle

Resynchronization is started for the relevant following spindle and commences as soon as the low-high edge of following interface signal is detected:

DB31, ... DBX31.4 (resynchronization)

The NC acknowledges the detection of the edge by outputting the NC/PLC interface signal:

DB31, ... DBX99.4 (synchronization running)

The interface signal "Synchronization running" is reset if:

- synchronization of the following spindle has been completed up to the stage at which there is synchronism at the setpoint end.
- the NST DB31, ... DBX31.4 (resynchronization) was reset.

Response of synchronous signals during additional movements for the following spindle

The superimposed component is calculated to establish the synchronism signals.

Example

```
Program code                                Comment
N51 SPOS=0 SPOS[2]=90
N52 OUPDEF(S2,S1,1,1,"FINE","DV")
N53 COUPON(S2,S1,77)
N54 M0                                      ; Offset=77°, "coarse", "fine"
                                              synchronous run signals exist.
N55 SPOS[2]=0 FA[S2]=3600                   ; Difference in speed, synchronism
                                              signals "coarse", "fine" are reported
N56 M0                                      ; (note tolerances, see above)
                                            ; Offset=0°, "coarse", "fine" synchronous
                                              run signals exist.
N60 M2=3 S2=500                             ; difference in speed, synchronism
                                              signals "coarse", "fine" are reported.
                                            ; offset undefined, synchronism signals
                                              "coarse", "fine" are reported.
N65 M0                                      ; (Note tolerances, see above)
```

Note

The axis enable signals can be canceled to interrupt a movement overlaid on the following spindle (e.g. SPOS). This component of the movement is not affected by IS "NC/PLC interface signal" DB31, ... DBX31.4 (resynchronization), but is restored by the REPOS operation.

Supplementary condition

NST DB31, ... DBX31.4 (resynchronization) only has any effect if there is a **defined offset position** between the following spindle and leading spindle.

This is the case following `COUPON` with offset positions such as `COUPON(...,77)` or `SPOS`, `SPOSA`, `M19` for the following spindle with a closed coupling..

14.4.3 Influence on synchronous operation via PLC interface

PLC interface signals

In synchronous operation, the influence of the PLC on the coupling resulting from the setting of LS and FS interface signals must be noted.

The effect of the main PLC interface signals on the synchronous spindle coupling is described below.

Spindle override (DB31, ... DBB19)

The spindle override value specified by the PLC in synchronous operation is only applied to the leading spindle.

Axis/spindle disable (DB31, ... DBX1.3)

The behavior of the axes involved can be found in the following table:

set: 1, disabled: 0

No.	LS/LA	FS/FA	Coupling	Response
1	0	0	OFF	Setpoints of axes are output
2	0	1	OFF	No setpoint output for FS/FA
3	1	0	OFF	No setpoint output for LS/LA
4	1	1	OFF	No setpoint output for LS/LA and FS/FA
5	0	0	ON	Setpoints of axes are output
6	0	1	ON	Disable not effective for FS/FA
7	1	0	ON	Disable also effective for FS/FA, no setpoint output
8	1	1	ON	No setpoint output for LS/LA and FS/FA

- This signal is no longer effective when the coupling for FS/FA is activated. → No. 6
- If the signal for the LS/LA is set, it also applies to the FS/FA(s) → No. 7
- A workpiece clamped between two spindles (workpiece transfer from front to rear-side machining) cannot be destroyed.

Servo enable (DB31, ... DBX2.1)

Cancellation of "Servo enable" for LS (either via PLC interface or internally in the control in the event of faults):

If the servo enable signal of the LS is set to "0" during synchronous operation and a setpoint coupling is active, a switchover to actual-value coupling is executed in the control. If the LS is in motion at this instant, it is decelerated to a standstill and an alarm message is generated. Synchronous operation remains active.

Cancellation of "Servo enable" for FS in synchronous operation (either via PLC interface or internally in the control in the event of faults):

If the "Servo enable" signal is not set for either of the spindles before synchronous operation is selected, synchronous operation is still activated when the coupling is switched on. The LS and FS however remain at a standstill until the servo enable signal is set for both of them.

Set "servo enable" for FS and LS:

When the signal edge of IS "Servo enable" = 1, the spindle either moves back to the old position (position on cancellation of servo enable) following the IS "Follow-up mode" (signal status = 0: Stop active) or the current positions (position offset) are used again (signal status = 1: Follow-up active).

Note

If the "servo enable" signal is canceled for the FS after Spindle STOP without the coupling being deactivated beforehand, then any synchronism error resulting from external intervention (e.g. manual rotation) will not be compensated when the "servo enable" signal is activated again.

This may result in the loss of the defined angular reference between the FS and LS for special applications. The programmed offset can be reproduced with the following interface signal:

IS "New synchronization" (DB31, ... DBX31.4).

Follow-up mode (DB31, ... DBX1.4)

The interface signal "Follow-up mode" is only relevant if the "servo enable" for the drive is canceled. Depending on the IS "follow-up mode," when "servo enable" is set for the FS and LS, either the spindle will return to the position recorded on cancellation of the servo enable signal (signal state = 0: Stop active) or the current positions will be used again (signal status = 1: Follow-up active).

Position measuring system 1/2 (DB31, ... DBX1.5 and 1.6)

Switchover between the position measuring systems for the FS and LS is not locked out in synchronous operation. The coupling is retained. It is however recommended that the measuring systems only be switched when synchronous mode is not active.

If "Park" status is selected for the FS or LS in synchronous operation, then the system responds as if "servo enable" had been canceled.

Delete distance to go/Spindle Reset (DB31, ... DBX2.2)

When Spindle reset is set for the LS in synchronous operation, the LS is decelerated to standstill at the selected acceleration rate. The FS and LS continue to operate in synchronous mode. The overlaid motion (except with COUP...) is terminated as quickly as possible.

Spindle stop (Feed stop) (DB31, ... DBX4.3)

When "Spindle stop" is set for the FS or LS, both coupled spindles are decelerated to standstill via a ramp, but continue to operate in synchronous mode.

As soon as IS "Spindle STOP" is no longer active for any of the spindles in the coupling, it is accelerated back up to the previous speed setpoint.

Application

"Spindle stop" can halt the synchronous spindle pair without offset since the servo loop position control remains operative.

Example

When the protective door is opened with an active synchronous spindle coupling, the FS and LS must be stopped without the coupling relationship being altered. This can be achieved by applying IS "Spindle stop" to halt the FS and LS (IS "Axis/spindle stationary" (DB31, ... DBX61.4) = 1). "Servo enable" can then be canceled for both spindles.

Delete S value (DB31, ... DBX16.7)

The S value programmed for the LS is deleted and the LS decelerated down to zero speed via a ramp. The FS and LS continue to operate in synchronous mode.

On the other hand, IS "Delete S value" has no affect on the FS in synchronous operation.

Resynchronize spindle 1/2 (DB31, ... DBX16.4 and 16.5)

It is also possible to synchronize the spindle (LS) with its positioning measuring system when it is operating in synchronous mode. It is however recommended that the leading spindle only be re-synchronized when synchronous mode is not active.

Re-synchronization (DB31, ... DBX31.4)

If the following and leading spindles have fallen out of synchronism, or failed to synchronize at all, the programmed offset can be restored using the following interface signal:

IS "Re-synchronization" (DB31, ... DBX31.4).

When the low-high edge of the VDI signal is detected, resynchronization is started for the following spindle in question and acknowledged by the NC with the following interface signal for the following spindle:

IS "Synchronization running" (DB31, ... DBX99.4) acknowledged for the following spindle.

Traverse keys for JOG (DB31, ... DBX4.6 and 4.7)

The "plus and minus traversing keys" for JOG are **not disabled** in the control for the FS in synchronous operation, i.e. the FS executes a superimposed motion if one of these keys is pressed.

Note

If superimposed traversing movements are to be precluded, they must be locked out by measures in the PLC user program.

NC Stop axes plus spindles (DB21, ... DBX7.4)

"NC Stop axes plus spindles" in synchronous operation decelerates the coupled spindles in accordance with the selected dynamic response. They continue to operate in synchronous mode.

NC Start (DB21, ... DBX7.1)

See section "Configuration of the behavior with NC start".

Note

NC Start after NC Stop does not deselect synchronous operation.

14.4.4 Differential speed between leading and following spindles

When does a differential speed occur?

A differential speed develops, e.g. with turning machine applications, when two spindles are opposite each other. Through the signed addition of two speed sources, a speed component is derived from the leading spindle via the coupling factor. In addition to this, it is possible to program the following for the following spindle:

- speed with S... and
- direction of rotation with M3, M4 or M5

The spindles can normally be synchronized by a coupling factor with the value '-1'. This sign reversal then results in a differential speed for the following spindle as compared to an additional programmed speed. This typical behavior in relation to the NC is illustrated in the following diagram.

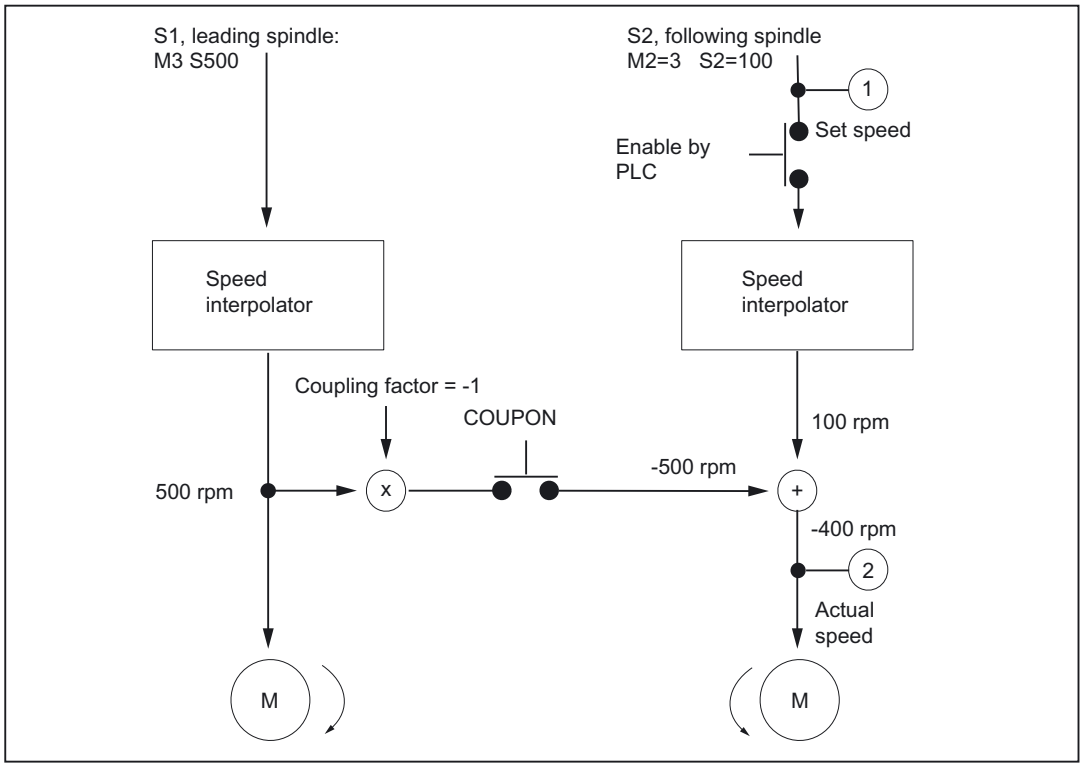


Figure 14-4 Schematic representation of process resulting in differential speed

Example

```
N01 M3 S500                ; S1 rotates positively at 500 rpm
                           ; the master spindle is spindle 1
N02 M2=3 S2=300            ; S2 rotates positively at 300 rpm
N05 G4 F1;
N10 COUPDEF(S2,S1,-1)      ; Coupling factor -1:1
N11 COUPON(S2,S1)          ; Activate coupling, the speed of following spindle S2
                           ; results from the speed of the main spindle S1 and
                           ; coupling factor
N26 M2=3 S2=100            ; Programming of difference in speed,
                           ; S2 is the following spindle
```

Application

Manufacturing operations with positioned leading spindle and rotating tools require exact synchronism with the counter spindle which then functions like a following spindle. A turret rotating about the following spindle allows parts to be machined with different tool types. The following diagram shows an application in which the tool is positioned parallel to the main spindle.

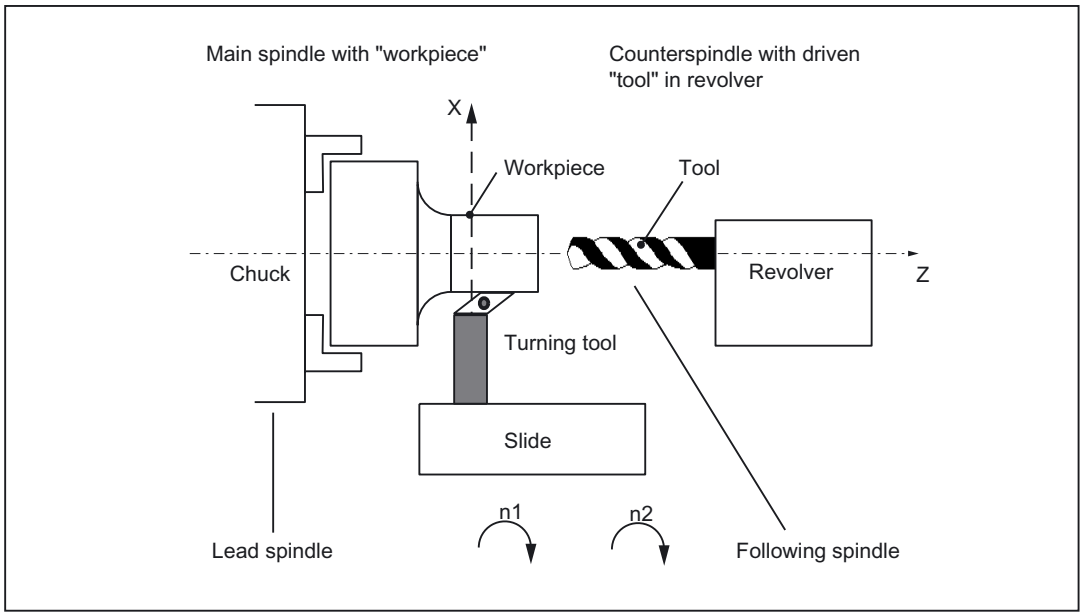


Figure 14-5 Application on a single-slide turning machine with turret about Z axis

Requirements

Basic requirements for differential speed programming:

- Synchronous spindle functionality is required.
- The dynamic response of the following spindle must be at least as high as that of the leading spindle. Otherwise, the system may suffer from reduced quality, for example, rigid tapping without a compensating chuck G331/G332.
- The differential speed must be programmed in the channel in which the following spindle is also configured. The leading spindle can be programmed in a different channel.
- The differential speed must be enabled for the following spindle by the PLC via IS "Enable overlaid movement" (DB31, ... DBX26.4). If the enable signal has not been set, alarm 16771 "Channel% Following axis% Overlaid movement not enabled" is output. This alarm is cleared when IS "Enable overlaid movement" (DB31, ... DBX26.4) is set or the coupling is terminated.

Note

The differential speed does not therefore affect the coupling process.

The following or leading spindle cannot change gear stages while a coupling is active.

Activate coupling with COUPONC

When the coupling is activated, the following spindle is accelerated, as before, to the leading spindle speed through application of the coupling factor. If the following spindle is already rotating (M3, M4) when the coupling is activated, it continues with this motion after coupling.

Deactivate coupling

If the coupling is deactivated, the following spindle continues to rotate at the speed corresponding to the sum of both speed components. The spindle behaves as if it had been programmed with the speed and direction transferred from the other spindle. When deactivating, there are no differences to the previous behavior.

Differential speed

A differential speed results from **renewed** programming of the following spindle (in the example S2=...) or M2=3, M2=4 in speed control mode **during** an active synchronous spindle coupling or by adopting the speed of the following spindle with COUONC.

Condition:

Speed S... must also be re-programmed with direction of rotation M3 or M4. Otherwise alarm 16111 "Channel% Block% Spindle% No speed programmed" is displayed.

Read offsets of following spindle

The current offset always changes when a differential speed is programmed. The current offset can be read at the setpoint end with $AA_COUP_OFFS[Sn] and at the actual value end with $VA_COUP_OFFS[Sn].

The last offset programmed returns the variable $P_COUP_OFFS[Sn].

Display differential speed

The programmed difference component is displayed as the speed setpoint for the programmed differential speed (in our example, corresponds to 100 rpm).

The actual speed refers to the motor speed. In the example, the actual speed is 500 rev/min * (-1) + 100 rpm = -400 rev/min.

IS NCK to PLC

Following spindle in speed-controlled operation

The IS "Spindle in setpoint range" (DB31, ... is set for the following spindle by the NCK if the programmed speed (see example above N26 with M2=3 S2=100) is reached. If a differential speed is programmed and not enabled by the PLC, this VDI interface signal is not set.

Even if a differential speed has been programmed, the following spindle remains under position control if this is required by the coupling.

Note

The axial VDI interface signal NCK → PLC IS "Superimposed motion" is set (DB31, ... DBX98.4) when setpoints in addition to the coupling setpoints are created by differential speed programming.

Actual direction of rotation CW (DB31, ... DBX83.7)

The IS "Actual direction of rotation CW" (DB31, ... DBX83.7) refers to the resultant direction of motor rotation.

IS PLC to NCK

Influence on following spindle via PLC interface

The effect of the axial VDI interface signals on the following spindle with differential speed in speed control mode is described below.

Delete distance to go/Spindle Reset (DB31, ... DBX2.2)

The programmed differential speed and direction of rotation can be terminated by IS "Delete distance-to-go/Spindle Reset" (DB31, ... DBX2.2). To delete the programmed speed only, it is possible to set IS "Delete S value" (DB31, ... DBX16.7).

Resynchronize spindle 1/2 (DB31, ... DBX16.4 and 16.5)

The IS "Resynchronize spindle 1/2" (DB31, ... DBX16.4/16.5) are **not** locked. Any positional offset is not compensated automatically by the coupling.

Invert M3/M4 (DB31, ... DBX17.6)

The IS "Invert M3/M4" (DB31, ... DBX17.6) only inverts the speed component additionally programmed for the following spindle.

The motion component generated by the synchronous spindle coupling remains unaffected.

Spindle override (DB31, ... DBB19)

The "Spindle override" VDI interface (DB31, ... DBB19) only impacts on the speed component additionally programmed for the following spindle. If the spindle override switch is transferred to all axial inputs, then any change in the spindle override value is applied **doubly** to the following spindle:

- once indirectly by a change in speed for the leading spindle and
- once in the programmed component of the following spindle.

The offset value can be adjusted accordingly in the PLC program.

Coupling deselection

If the coupling is deactivated, the following spindle continues to rotate at the speed corresponding to the sum of both speed components. The motion transition upon coupling deselection is at continuous speed.

With COUPOF, the spindle behaves as if it had been programmed with the speed and direction transferred from the other spindle. In the example, this would be M4 S400.

When COUPOFS is programmed, the following spindle is decelerated to standstill from the current speed.

Activate additional functions

The following spindle can also be a master spindle. In this case, it is capable of additional functions.

- Rotational feedrate with G95, G96 and G97. With G96 S2=... the "constant cutting speed" can be activated for the following spindle.

  The speed dependent on the position of the transverse axis is the setpoint speed for the speed interpolator of spindle 2 and is therefore included in the total speed of S2.
- Rigid tapping without compensating chuck with G331, G332.

14.4.5 Behavior of synchronism signals during synchronism correction

Effect of synchronism correction

New synchronism signals are produced by comparing the actual values with the corrected setpoints. Once a correction process has been undertaken, the synchronism signals should be present again.

14.4.6 Delete synchronism correction and NC reset

Variable $AA_COUP_CORR[Sn] returns the value zero for different situations in which the synchronism correct is deleted:

- Once a synchronized spindle coupling has been activated for the following in question with COUPON(..)/COUPONC(..), an existing synchronism correction is adopted in the setpoint position.
- A synchronism correction active during NC reset but not at the parts program end is adopted in the setpoint position. This does not affect the synchronism signals.
- At M30, an existing synchronism correction is retained
- At the user end, the correction value can also be deleted at any early point by describing the variable $AA_COUP_CORR with the **value zero**. The synchronism correction is removed immediately and using a ramp with reduced acceleration rate if larger values are involved.

14.4.7 Special points regarding start-up of a synchronous spindle coupling

Spindle start-up

The leading and following spindles must be started up initially like a normal spindle. The corresponding method is described in
References:
/IADC/ Commissioning Manual SINUMERIK 840D
/FB1/ Function Manual Basic Functions, Spindles (S1)

Requirements

The following parameters must then be set for the synchronous spindle pair:

- The machine numbers for the leading and following spindles

  (for permanently configured coupling with channel-specific machine data MD21300 $MC_COUPLE_AXIS_1[n])
- required coupling mode (setpoint, actual value or speed coupling)

  (for permanently configured coupling with channel-specific machine data MD21310 $MC_COUPLING_MODE_1[n])
- select the gear stage(s) of FS and LS for synchronous operation
- The following coupling properties are still applicable for permanently configured synchronous spindle coupling:
  - Block change response in synchronous spindle operation:

    MD21320 $MC_COUPLE_BLOCK_CHANGE_CTRL_1
  - Coupling cancellation response:

    MD21330 $MC_COUPLE_RESET_MODE_1
  - Write-protection for coupling parameters:

    MD21340 $MC_COUPLE_IS_WRITE_PROT_1
  - Translation parameters for synchronized spindle coupling:

    SD42300 $SC_COUPLE_RATIO_1[n]

Command behavior of FS and LS for setpoint coupling

In order to obtain the best possible synchronism in **setpoint couplings**, the FS and LS must have the same dynamic response as the response to setpoint changes. The axial control loops (position, speed and current controllers) should each be set to the **optimum** value so that variances can be eliminated as quickly and efficiently as possible.

The **dynamic response adaptation** function in the setpoint branch is provided to allow differing dynamic responses of axes to be matched without loss of control quality. The following control parameters must each be set optimally for the FS and LS:

- K_V factor (MD32200 $MA_POSCTRL_GAIN)
- Feedforward control parameters

  MD32620 $MA_FFW_MODE

  MD32610 $MA_VELO_FFW_WEIGHT

  MD32650 $MA_AX_INERTIA

  MD32800 $MA_EQUIV_CURRCTRL_TIME

  MD32810 $MA_EQUIV_SPEEDCTRL_TIME

References:
/FB2/ Function Manual Extended Functions; Compensations (K3)

Behavior during loss of synchronism:

- axis-specific MD32620 $MA_FFW_MODE

We would recommend setting the **feedforward control mode** of the following axis to speed feedforward control with Tt symmetrization MD32620 = 3.

This feedforward control mode can be further optimized for a more secure symmetrization process by changing the axis-specific machine data:

Machine data	Description
MD32810 EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
MD37200 COUPLE_POS_TOL_COURSE	Threshold value for "Coarse synchronism"
MD37210 COUPLE_POS_TOL_FINE	Threshold value for "Fine synchronism"
MD37220 COUPLE_VELO_TOL_COURSE	Velocity tolerance 'coarse'
MD37220 COUPLE_VELO_TOL_FINE	Velocity tolerance 'fine'

In such cases, higher threshold values for the synchronism signals and larger position and/or speed tolerances result in more stable results.

Dynamic response adaptation

The FS and the coupled LS must have the same dynamic response to setpoint changes. The same dynamic response means that their following errors must be equal at any given speed.

The dynamic response adaptation function in the setpoint branch makes it possible to obtain an excellent match in the response to setpoint changes between axes, which have different dynamic characteristics (control loops). The difference in the equivalent time constants between the dynamically "weakest" spindle and the other spindle in the coupling must be entered as the dynamic response adaptation time constant.

Example

When the speed feedforward control is active, the dynamic response is primarily determined by the equivalent time constant of the "slowest" speed control loop.

Lead spindle:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME [n] = 5 ms

Following spindle:

MD32810 $MA_EQUIV_SPEEDCTRL_TIME [n] = 3 ms

Time constant of dynamic response adaptation for the following spindle:

MD32910 $MA_DYN_MATCH_TIME [n] = 5 ms - 3 ms = 2 ms

The dynamic response adaptation must be activated axially via MD32900 $MA_DYN_MATCH_ENABLE.

The dynamic adaptation setting can be checked by comparing the following errors of the FS and LS (in Diagnosis operating area; Service Axes display). Their following errors must be identical when they are operating at the same speed!

For the purpose of fine tuning, it may be necessary to adjustservo gain factors or feedforward control parameters slightly to achieve an optimum result.

Control parameter sets

A separate parameter set with servo loop setting is assigned to each gear stage on spindles.

These parameter sets can be used, for example, to adapt the dynamic response of the leading spindle to the following spindle in synchronous operation.

When the coupling is deactivated (speed or positioning mode), it is therefore possible to select other position controller parameters for the FS and LS. To utilize this option, a separate gear stage must be used for synchronous operation and selected before synchronous mode is activated.

The coupling parameters of the following spindle can be set as follows using this machine data:

MD30455 $MA_MISC_FUNCTION_MASK

Bit 5=0: Synchronized spindle coupling, following spindle:

Position control, feedforward control and parameter block are set for the following spindle.

Bit5=1: Synchronous spindle coupling:

The control parameters of the following spindle are set as in an uncoupled scenario.

The following control parameters must be set identically for the FS and LS:

- Fine interpolator type (MD33000 $MA_FIPO_TYPE)
- Axial jerk limitation

  MD32400 $MA_AX_JERK_ENABLE

  MD32410 $MA_AX_JERK_TIME

  MD32420 $MA_JOG_AND_POS_JERK_ENABLE

  MD32430 $MA_JOG_AND_POS_MAX_JERK

References:

/FB1/ Function Manual, Basic Functions, Velocities, Setpoint-Actual Value Systems, Closed-Loop Control (G2)

Separate dynamic response for spindle and axis operations

In spindle and axis operations, dynamic programming FA, OVRA, ACC and VELOLIMA can be set separately from one another with the following MD:

MD30455 $MA_MISK_FUNCTION_MASK Bit 6=0

Assignment is undertaken by the programmed axis or spindle identifier. E.g., in spindle operation, VELOLIMA[S1]=50 therefore only reduces the maximum speed to 50% and in axis operation, VELOLIMA[C]=50 only reduces the maximum speed to 50%.

If e.g. B. VELOLIMA[S1]=50 and VELOLIMA[C]=50 are to have the same effect as before with this machine data, the programming of FA, OVRA, ACC and VELOLIM have an effect regardless of the programmed identifiers:

MD30455 $MA_MISK_FUNCTION_MASK Bit 6=1

Knee-shaped acceleration characteristic

For the leading spindle, the effect of a knee-shaped acceleration characteristic on the following spindle is identified by the following axis-specific machine data:

MD35220 $MA_ACCEL_REDUCTION_SPEED_POINT (speed for reduced acceleration) and

MD35230 $MA_ACCEL_REDUCTION_FACTOR (reduced acceleration).

If MD35242 $MA_ACCEL_REDUCTION_TYPE is present, it is also used to configure the type of acceleration reduction. Otherwise a hyperbolic drop in acceleration is assumed.

If the dynamic response of a following spindle is lower than that of the leading spindle when the coupling factor is taken into account, the leading spindle dynamic response is reduced to the required level while the coupling is active.

The acceleration should be **constant** over the entire speed range for the following spindle. However, if a knee-shaped acceleration characteristic is also stored in the above-mentioned machine data for the following spindle, this is only taken into account when the spindles are coupled in. The setpoints of the following spindle are applied for the specified knee-shaped acceleration characteristic.

References:

/FB1/ Function Manual, Basic Functions, Acceleration (B2), Section Knee-shaped acceleration characteristic

Actual value coupling

In an actual-value coupling (AV), the drive for the FS must be considerably more dynamic than the leading spindle drive. The individual drives in an actual-value coupling are also set optimally according to their dynamic response.

An actual-value coupling should only be used in exceptional cases.

Speed coupling

The velocity coupling (VV) corresponds internally to a setpoint coupling (DV), but with lower dynamic requirements of the FS and LS. A position control servo loop is not needed for the FS and/or LS. Measuring systems are not required.

Threshold values for coarse/fine synchronism

After controller optimization and feedforward control setting, the threshold values for coarse and fine synchronism must be entered for the FS.

- Threshold value for "Coarse synchronism"

  Axis-specific MD7200: AV, DV: COUPLE_POS_TOL_COARSE

  MD37220: VV: COUPLE_VELO_TOL_COARSE

- Threshold value for "Fine synchronism"

  Axis-specific MD37210: AV, DV: COUPLE_POS_TOL_FINE

  MD37230: VV: COUPLE_VELO_TOL_FINE

The values of the FS must be calculated according to the accuracy requirements of the machine manufacturer and the PLC interface must be checked via the service display.

Angular offset LS/FS

If there must be a defined angular offset between the FS and LS, e.g. when synchronous operation is activated, the "zero degree positions" of the FS and LS must be mutually adapted. This can be done with the following machine data:

- MD34100 $MA_REFP_SET_POS
- MD34080 $MA_REFP_MOVE_DIST
- MD34090 $MA_REFP_ MOVE_DIST_CORR

References:

/FB1/ Function Manual, Basic Functions, Reference Point Approach (R1)

Service display for FS

The following values are displayed for the following spindle for start-up in synchronous operation in the "Service Values Axes" display in the "Diagnosis" operating area:

- Actual deviation between setpoints of FS and LS

  Value displayed: Position offset in relation to leading spindle (setpoint)

  (value corresponds to angular offset between FS and LS that can be read with axis variable $AA_COUP_OFFS in the parts program)

- Actual deviation between actual values of FS and LS

  Value displayed: Position offset in relation to leading spindle (actual value)

References:

/BAD, BEM/ HMI Advanced/Embedded, Service display, Section "Service drive"

14.5 Examples

Programming example

```
                                    ; Leading spindle = master spindle =
                                    ; Following spindle = spindle 2
N05 M3 S3000 M2=4 S2=500            ; Master spindle rotates at 3000 rpm
                                    ; FS: 500/min.
N10 COUPDEF (S2, S1, 1, 1, "No",    ; Def. of coupling, can also
"Dv")                               ; be configured
N70 SPCON                           ; Include leading spindle in position control
                                    ; (setpoint value coupling).
N75 SPCON(2)                        ; Include slave spindle in position control
N80 COUPON (S2, S1, 45)             ; On-the-fly coupling to offset position =
                                      45 degrees
N200 FA [S2] = 100                  ; Positioning speed = 100 degrees/min
N205 SPOS[2] = IC(-90)              ; Traverse with 90° overlay in negative
                                      direction
N210 WAITC(S2, "Fine")              ; Wait for "fine" synchronism
N212 G1 X.., Y.. F...               ; Machining
N215 SPOS[2] = IC(180)              ; Traverse with 180° overlay in positive
                                      direction
N220 G4 S50                         ; Dwell time = 50 revolutions
                                    ; of master spindle
N225 FA [S2] = 0                    ; Activate configured speed (MD)
N230 SPOS[2] = IC (-7200)           ; 20 rev. with configured velocity
                                    ; in neg. direction
N350 COUPOF (S2, S1)                ; On-the-fly decoupling, S = S2 = 3000
N355 SPOSA[2] = 0                   ; Stop slave spindle at zero degrees
N360 G0 X0 Y0
N365 WAITS(2)                       ; Wait for spindle 2
N370 M5                             ; Stop slave spindle
N375 M30
```

14.6 Data lists

14.6.1 Machine data

14.6.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10000	AXCONF_MACHAX_NAME_TAB	Machine axis name

14.6.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20070	AXCONF_MACHAX_USED	Machine axis number valid in channel
21300	COUPLE_AXIS_1	Definition of synchronous spindle pair
21310	COUPLING_MODE_1	Type of coupling in synchronous spindle mode
21320	COUPLE_BLOCK_CHANGE_CTRL_1	Block change behavior in synchronous spindle operation
21330	COUPLE_RESET_MODE_1	Coupling abort behavior
21340	COUPLE_IS_WRITE_PROT_1	Coupling parameters are write-protected

14.6.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30455	MISK_FUNCTION_MASK	Axis functions
30550	AXCONF_ASSIGN_MASTER_CHAN	Reset position of channel for axis change
32200	POSCTRL_GAIN	Servo gain factor
32400	AX_JERK_ENABLE	Axial jerk limitation
32410	AX_JERK_TIME	Time constant for axial jerk filter
32420	JOG_AND_POS_JERK_ENABLE	Initial setting for axial jerk limitation
32430	JOG_AND_POS_MAX_JERK	Axial jerk
32610	VELO_FFW_WEIGHT	Feedforward control factor for speed feedforward control
32620	FFW_MODE	Feedforward control mode
32650	AX_INERTIA	Moment of inertia for torque feedforward control
32800	EQUIV_CURRCTRL_TIME	Equivalent time constant current control loop for feedforward control
32810	EQUIV_SPEEDCTRL_TIME	Equivalent time constant speed control loop for feedforward control
34080	REFP_MOVE_DIST	Reference point approach distance

Number	Identifier: $MA_	Description
34090	REFP_MOVE_DIST_CORR	Reference point offset
34100	REFP_SET_POS	Reference point value
35000	SPIND_ASSIGN_TO_MACHAX	Assignment of spindle to machine axis
37200	COUPLE_POS_TOL_COARSE	Threshold value for "Coarse synchronism"
37210	COUPLE_POS_TOL_FINE	Threshold value for "Fine synchronism"
37220	COUPLE_VELO_TOL_COARSE	Speed tolerance "coarse" between leading and following spindles
37230	COUPLE_VELO_TOL_FINE	Speed tolerance "fine" between leading and following spindles

14.6.2 Setting data

14.6.2.1 Channel-specific setting data

Number	Identifier: $SC_	Description
42300	COUPLE_RATIO_1	Transmission parameters for synchronous spindle operation

14.6.3 Signals

14.6.3.1 Signals to channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
NC start	DB21, ...DBX7.1	DB3200.DBX7.1
NC stop axes plus spindle	DB21, ...DBX7.4	DB3200.DBX7.4

14.6.3.2 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Dry run feedrate selected	DB21, ...DBX24.6	DB1700.DBX0.6
Feedrate override selected for rapid traverse	DB21, ...DBX25.3	DB1700.DBX1.3

14.6.3.3 Signals to axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Axis/spindle disable	DB31, ...DBX1.3	DB380x.DBX1.3
Follow-up mode	DB31, ...DBX1.4	DB380x.DBX1.4

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Position measuring system 1, position measuring system 2	DB31, ...DBX1.5/6	DB380x.DBX1.5/6
Controller enable	DB31, ...DBX2.1	DB380x.DBX2.1
Distance-to-go/Spindle RESET	DB31, ...DBX2.2	DB380x.DBX2.2
Spindle stop/feed stop	DB31, ...DBX4.3	DB380x.DBX4.3
Traversing keys for JOG	DB31, ...DBX4.6/7	DB380x.DBX4.6/7
Re-synchronize spindle 1, re-synchronize spindle 2	DB31, ...DBX16.4/5	DB380x.DBX2000.4/5
Delete S value	DB31, ...DBX16.7	DB380x.DBX2000.7
Feedrate override valid	DB31, ...DBX17.0	DB380x.DBX2001.0
Invert M3/M4	DB31, ...DBX17.6	DB380x.DBX2001.6
Spindle override	DB31, ...DBX19	DB380x.DBB2003
Re-synchronizing	DB31, ...DBX31.4	-
Disable synchronization	DB31, ...DBX31.5	-

14.6.3.4 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1, referenced/synchronized 2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Synchronous mode	DB31, ...DBX84.4	DB390x.DBX2002.4
Synchronism fine	DB31, ...DBX98.0	-
Synchronism coarse	DB31, ...DBX98.1	-
Actual value coupling	DB31, ...DBX98.2	-
Superimposed motion	DB31, ...DBX98.4	DB390x.DBX5002.4
Leading spindle LS/LA active	DB31, ...DBX99.0	-
Following spindle FS/FA active	DB31, ...DBX99.1	-

14.6.4 System variables

14.6.4.1 System variables

System variables	Description
$P_COUP_OFFS[following spindle]	Programmed offset of the synchronous spindle
$AA_COUP_OFFS[following spindle]	Position offset for synchronous spindle (setpoint)
$VA_COUP_OFFS[following spindle]	Position offset for synchronous spindle (actual value)

A more detailed description of system variables can be found in
References:
/PGA1/, Parameter Manual System Variables

S7: Memory configuration

15

15.1 Brief description

Memory types

To store and manage data, the NC requires a static memory and a dynamic memory:

- **Static NC memory**

  In the static NC memory, the program data (part programs, cycles, etc.) and the current system and user data (tool management, global user data, etc.) is saved to **persistent memory**.

- **Dynamic NC memory**

  In the dynamic NC memory the data is saved to **non-persistent memory**. The data here concerns information generated by the NC that is only required for a limited time (e.g., macros, local user data, interpolation buffer, etc.).

Memory organization

The memory areas of the individual data groups in the static and dynamic NC memories have defined sizes, which are fixed when the memory is configured.

This type of memory organization ensures the **deterministic** behavior of the control:
The reserved memory area is guaranteed throughout part program processing.

Memory configuration

When booted for the first time, the system enters default values in all other memory-configuration machine data. This configuration is adequate in most cases. However, the machine manufacturer/user can make changes at any time (**Reconfiguration**).

15.2 Memory organization

15.2.1 Active and passive file system

The static NC memory contains an active and passive file system.

Active file system

The active file system contains system data used to parameterize the NCK:

- Machine data
- Setting data
- Option data
- Global user data (GUD)
- Tool-offset/magazine data
- Protection zones
- R parameters
- Work offsets/FRAME
- Sag compensations
- Quadrant error compensation
- Leadscrew error compensation

This data represents the current work data of the NCK.

The user's view of the active file system is data-oriented.

Passive file system

The passive file system contains all files loaded onto the NCK:

- Main programs
- Subprograms
- Global-user-data and macro definition files (*.DEF)
- Standard cycles
- User cycles
- Workpieces
- Comments

The user's view of the passive file system is file-oriented.

15.2.2 Reconfiguration

Reconfiguration

The following actions result in the reconfiguration of the static and/or dynamic NC memory:

- Changing the settings of memory-configuration machine data:
  MD... $..._**MM**_...
- Changing the number of channels

Protecting against loss of data

NOTICE
A reconfiguration of the **static** NC memory results in a loss of data on the active and passive file system. Before activating the modified memory configuration, you must must first save the data by creating a **series machine start-up file**.

15.3 Configuration of the static user memory

15.3.1 Division of the static NC memory

The figure below shows the division of the static NC memory for SINUMERIK 840D sl:

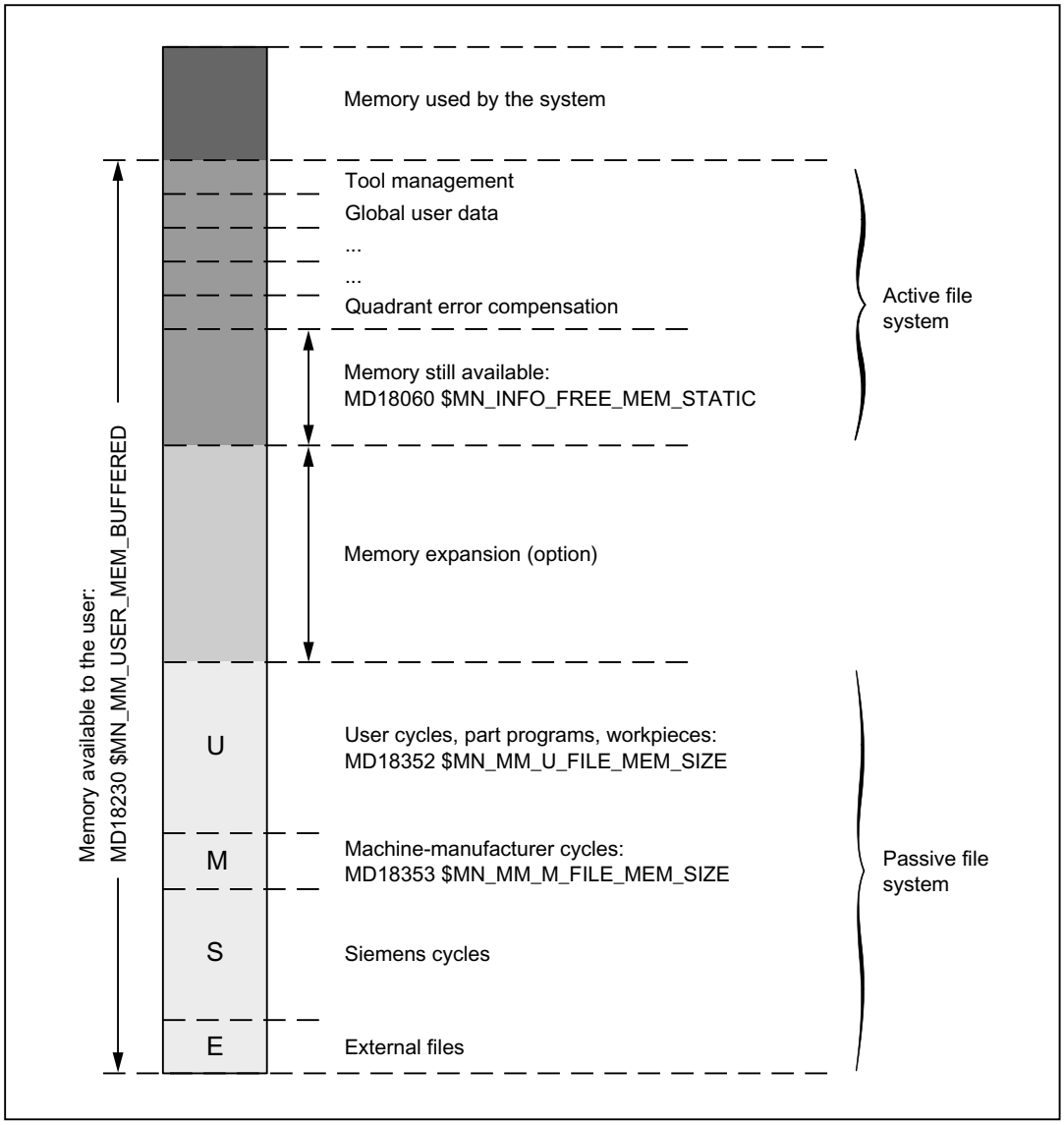


Figure 15-1 Static NC memory for SINUMERIK 840D sl

Static user memory

The static NC memory is used jointly by the system and by the user.

The area available to the user is defined as the static user memory. It contains the data from the active and passive file system.

Static-user-memory size

The size of the static user memory is defined in machine data:

MD18230 $MN_MM_USER_MEM_BUFFERED

Memory space for passive file system

The memory space for the passive file system has a defined size and is divided into the following partitions:

Partition	Storage of:
S (**S**iemens = Control manufacturer)	Files from the _N_CST_DIR directory (Siemens cycles)
M (**M**anufacturer = Machine manufacturer)	Files from the _N_CMA_DIR directory (Machine-manufacturer cycles)
U (**U**ser = End customer)	Files from the _N_CUS_DIR directory (User cycles, part programs, workpieces)
E (**E**XT = External)	External files (e.g., part programs from USB FlashDrive or CD). **Note:** To avoid naming conflicts with internal part programs, part programs installed from external data carriers are stored in a separate directory (_N_EXT_DIR).

Advantage of separation:

This separation into different partitions ensures that a series machine start-up file can be reloaded onto the NCK even in the event of an NCK software or cycle package upgrade (that has expanded the area of Siemens cycles).

Size of the partitions:

The size of partitions **S** and **E** are preset and cannot be modified.

You can divide the remaining memory available for the passive file system into the partitions **U** and **M** as you see fit. The settings are made with machine data:

MD18352 $MN_MM_U_FILE_MEM_SIZE
(end-user memory for part programs/cycles/files)

MD18353 $MN_MM_M_FILE_MEM_SIZE
(memory size for cycles/files of the machine manufacturer)

The maximum adjustable values depend on:

- The system and thus the memory space available (including an optional memory expansion)
- The defined maximum values (see "Detailed MD description")

Memory space for active file system

The memory space for the active file system is divided into various data areas (tool management, global user data, etc.), which can be defined individually using machine data.

The memory still available is shown in machine data:

MD18060 $MN_INFO_FREE_MEM_STATIC
(free-static-memory display data)

You can expand the sizes of the individual memory areas for the active file system with the relevant machine data until the available memory has been used.

Note

The memory required to expand the memory areas is displayed in the "Startup" area of the user interface. This information enables the system startup engineer to estimate the actual memory requirements for the planned expansion.

Memory expansion (option)

The machine manufacturer can also acquire additional static user memory ("Expansion of buffered CNC user memory" option).

You can use the additional memory space as required to expand partitions U and M or to expand the memory area of the active file system.

15.3.2 Startup

Procedure

1. Load standard machine data.
2. If you have selected the "Expansion of buffered CNC user memory" option:

   Activate the option.
3. Preset machine data:
   MD18230 $MN_MM_USER_MEM_BUFFERED
   with a high value (> default memory available + optional additional memory).
4. Reset the NCK.

   Alarm 6030 "User memory limit adjusted" is triggered and the maximum memory available for the user is entered in MD18230 (including optional memory expansion). Default values are entered in all other memory-configuration machine data.

5. Set the sizes of partitions U and M in machine data:

   MD18352 $MN_MM_U_FILE_MEM_SIZE
   (end-user memory for part programs/cycles/files)

   MD18353 $MN_MM_M_FILE_MEM_SIZE
   (memory size for cycles/files of the machine manufacturer)

6. Activate the number of required channels and axes.

   References:

   SINUMERIK 840D Commissioning Manual
   SINUMERIK 840Di Manual

7. You can adjust the default memory division by increasing/decreasing individual memory areas of the active file system (tool management, global user data, etc.) for each user.
   - The static user memory still available is displayed in machine data:

     MD18060 $MN_INFO_FREE_MEM_STATIC
     (free-static-memory display data)

   - Setting machine data (→ Data lists).

     References:
     Detailed machine-data description

8. Reset the NCK.

   The memory is set up again.

15.4 Configuration of the dynamic user memory

15.4.1 Division of the dynamic NC memory

The figure below shows the division of the dynamic NC memory:

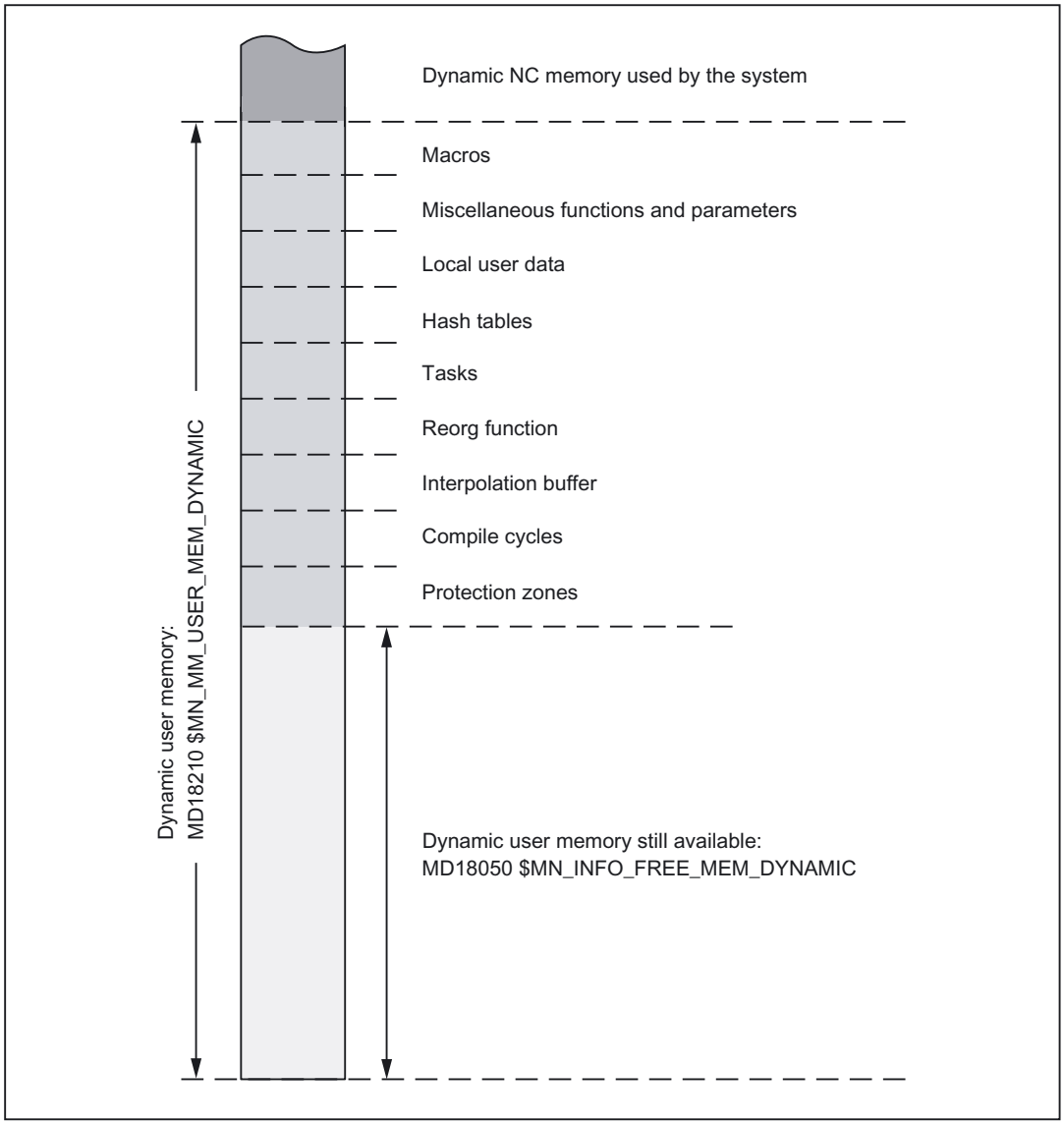


Figure 15-2 Dynamic NC memory

Dynamic user memory

The dynamic NC memory is used jointly by the system and by the user. The area available to the user is defined as the dynamic user memory.

Dynamic-user-memory size

The size of the dynamic user memory is set in machine data:

MD18210 $MN_MM_USER_MEM_DYNAMIC

Changes are not usually required as an appropriate value is automatically set during the reconfiguration.

Dynamic user memory still available

The dynamic memory still available is shown in machine data:

MD18050 $MN_INFO_FREE_MEM_DYNAMIC
(free-dynamic-memory display data)

The content of this machine data indicates how much memory is available to expand the user data areas (local user data, IPO buffer, etc.) for each channel.

15.4.2 Startup

You can adjust the default memory division by increasing/decreasing individual memory areas for each user.

Procedure

1. The dynamic user memory still available is displayed in machine data:

   MD18050 $MN_INFO_FREE_MEM_DYNAMIC
   (free-dynamic-memory display data)

2. Setting machine data (→ Data lists).

   References:
   Detailed machine-data description

3. Reset the NCK.

   The memory is set up again.

15.5 Data lists

15.5.1 Machine data

15.5.1.1 General machine data

Number	Identifier: $MN_	Description
10134	MM_NUM_MMC_UNITS	Number of simultaneous HMI communication partners
10850	MM_EXTERN_MAXNUM_OEM_GCODES	Maximum number of OEM-G codes
10880	MM_EXTERN_CNC_SYSTEM	Definition of the control system to be adapted
10881	MM_EXTERN_GCODE_SYSTEM	ISO_3 Mode: GCodeSystem
18050	INFO_FREE_MEM_DYNAMIC	Free-dynamic-memory display data
18060	INFO_FREE_MEM_STATIC	Free-static-memory display data
18070	INFO_FREE_MEM_DPR	Display data for free memory in dual-port RAM
18072	INFO_FREE_MEM_CC_MD	Displays memory available in the CC-MD memory
18078	MM_MAX_NUM_OF_HIERARCHIES	Maximum number of definable magazine-location-type hierarchies
18079	MM_MAX_HIERARCHY_ENTRIES	Maximum permissible number of entries in a magazine-location-type hierarchy
18080	MM_TOOL_MANAGEMENT_MASK	Mask for reserving memory for tool management
18082	MM_NUM_TOOL	Number of tools managed by NCK
18084	MM_NUM_MAGAZINE	Number of magazines managed by NCK
18086	MM_NUM_MAGAZINE_LOCATION	Number of magazine locations
18088	MM_NUM_TOOL_CARRIER	Maximum number of definable toolholders
18090	MM_NUM_CC_MAGAZINE_PARAM	Compile-cycle tool management: quantity of magazine data
18092	MM_NUM_CC_MAGLOC_PARAM	Compile-cycle tool management: quantity of magazine-location data
18094	MM_NUM_CC_TDA_PARAM	Compile-cycle tool management: quantity of TDA data
18096	MM_NUM_CC_TOA_PARAM	Compile-cycle tool management: quantity of TOA data
18098	MM_NUM_CC_MON_PARAM	Compile-cycle tool management: quantity of monitor data
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Number of tool offsets in NCK
18102	MM_TYPE_OF_CUTTING_EDGE	Type of D-number programming
18104	MM_NUM_TOOL_ADAPTER	Tool adapter in TO area
18105	MM_MAX_CUTTING_EDGE_NO	Maximum value of D number
18106	MM_MAX_CUTTING_EDGE_PERTOOL	Maximum number of D numbers per tool
18108	MM_NUM_SUMCORR	Additive offsets in the TO area
18110	MM_MAX_SUMCORR_PER_CUTTEDGE	Maximum number of sum offsets per cutting edge

Number	Identifier: $MN_	Description
18112	MM_KIND_OF_SUMCORR	Properties of additive offsets in the TO area
18114	MM_ENABLE_TOOL_ORIENT	Assign orientation to cutting edges
18116	MM_NUM_TOOL_ENV	Number of tool environments in TO area
18118	MM_NUM_GUD_MODULES	Number of GUD modules
18120	MM_NUM_GUD_NAMES_NCK	Number of global user variables
18130	MM_NUM_GUD_NAMES_CHAN	Number of channel-specific user variables
18140	MM_NUM_GUD_NAMES_AXIS	Number of axis-specific user variables
18150	MM_GUD_VALUES_MEM	Memory space for global user variables
18160	MM_NUM_USER_MACROS	Number of macros
18170	MM_NUM_MAX_FUNC_NAMES	Number of miscellaneous functions
18180	MM_NUM_MAX_FUNC_PARAM	Number of additional parameters
18190	MM_NUM_PROTECT_AREA_NCK	Number of protection zones in NCK
18200	MM_NUM_CCS_MAGAZINE_PARAM	Number of Siemens OEM magazine data
18201	MM_TYPE_CCS_MAGAZINE_PARAM	Siemens OEM magazine data type
18202	MM_NUM_CCS_MAGLOC_PARAM	Number of Siemens OEM magazine location data
18203	MM_TYPE_CCS_MAGLOC_PARAM	Siemens OEM magazine location data type
18204	MM_NUM_CCS_TDA_PARAM	Number of Siemens OEM tool data
18205	MM_TYPE_CCS_TDA_PARAM	Siemens OEM tool data type
18206	MM_NUM_CCS_TOA_PARAM	Number of Siemens OEM data per cutting edge
18207	MM_TYPE_CCS_TOA_PARAM	Siemens OEM data type per cutting edge
18208	MM_NUM_CCS_MON_PARAM	Number of Siemens OEM monitor data
18209	MM_TYPE_CCS_MON_PARAM	Siemens OEM monitor data type
18210	MM_USER_MEM_DYNAMIC	User memory in DRAM
18220	MM_USER_MEM_DPR	User memory in dual-port RAM
18230	MM_USER_MEM_BUFFERED	User memory in SRAM
18231	MM_USER_MEM_BUFFERED_TYPEOF	Data-buffer technology
18232	MM_ACTFILESYS_LOG_FILE_MEM	System: Log-file size
18238	MM_CC_MD_MEM_SIZE	Machine-data compile cycles in SRAM
18240	MM_LUD_HASH_TABLE_SIZE	Hash-table size for user variables
18242	MM_MAX_SIZE_OF_LUD_VALUE	Maximum LUD-variable array size
18250	MM_CHAN_HASH_TABLE_SIZE	Hash-table size for channel-specific data
18260	MM_NCK_HASH_TABLE_SIZE	Hash-table size for global data
18270	MM_NUM_SUBDIR_PER_DIR	Number of subdirectories
18280	MM_NUM_FILES_PER_DIR	Number of files per directory
18290	MM_FILE_HASH_TABLE_SIZE	Hash-table size for files in a directory
18300	MM_DIR_HASH_TABLE_SIZE	Hash-table size for subdirectories
18310	MM_NUM_DIR_IN_FILESYSTEM	Number of directories in passive file system
18320	MM_NUM_FILES_IN_FILESYSTEM	Number of files in passive file system
18332	MM_FLASH_FILE_SYSTEM_SIZE	Size of flash file system on PCNC
18342	MM_CEC_MAX_POINTS	Maximum table size for sag compensation

Number	Identifier: $MN_	Description
18350	MM_USER_FILE_MEM_MINIMUM	Minimum part-program memory
18352	MM_U_FILE_MEM_SIZE	End-user memory for part programs/cycles/files
18353	MM_M_FILE_MEM_SIZE	Memory size for cycles/files of the machine manufacturer
18354	MM_S_FILE_MEM_SIZE	Memory size for cycles/files of the NC manufacturer
18355	MM_T_FILE_MEM_SIZE	Memory size for temporary files
18356	MM_E_FILE_MEM_SIZE	Memory size for external files
18360	MM_EXT_PROG_BUFFER_SIZE	FIFO buffer size for execution from external source (DRAM)
18362	MM_EXT_PROG_NUM	Number of program levels that can be processed simultaneously from external
18370	MM_PROTOC_NUM_FILES	Maximum number of log files
18371	MM_PROTOC_NUM_ETPD_STD_LIST	Number of standard ETPD data lists
18372	MM_PROTOC_NUM_ETPD_OEM_LIST	Number of ETPD OEM data lists
18373	MM_PROTOC_NUM_SERVO_DATA	Number of servo data for log
18374	MM_PROTOC_FILE_BUFFER_SIZE	Log-file buffer size
18375	MM_PROTOC_SESS_ENAB_USER	User-enabled for sessions
18390	MM_COM_COMPRESS_METHOD	Supported compression method
18400	MM_NUM_CURVE_TABS	Number of curve tables (SRAM)
18402	MM_NUM_CURVE_SEGMENTS	Number of curve segments (SRAM)
18403	MM_NUM_CURVE_SEG_LIN	Number of linear curve segments (SRAM)
18404	MM_NUM_CURVE_POLYNOMS	Number of curve table polynomials (SRAM)
18406	MM_NUM_CURVE_TABS_DRAM	Number of curve tables (DRAM)
18408	MM_NUM_CURVE_SEGMENTS_DRAM	Number of curve segments (DRAM)
18409	MM_NUM_CURVE_SEG_LIN_DRAM	Number of linear curve segments (DRAM)
18410	MM_NUM_CURVE_POLYNOMS_DRAM	Number of curve table polynomials (DRAM)
18450	MM_NUM_CP_MODULES	Maximum number of CP modules
18452	MM_NUM_CP_MODUL_LEAD	Maximum number of leading values per CP coupling module
18500	MM_EXTCOM_TASK_STACK_SIZE	Stack size of external communication task (DRAM)
18502	MM_COM_TASK_STACK_SIZE	Stack size in Kbytes of communication task (DRAM)
18510	MM_SERVO_TASK_STACK_SIZE	Stack size of servo task (DRAM)
18512	MM_IPO_TASK_STACK_SIZE	Stack size of IPO task (DRAM)
18520	MM_DRIVE_TASK_STACK_SIZE	Stack size of drive task (DRAM)
18540	MM_PLC_TASK_STACK_SIZE	Stack size of PLC task (DRAM)
18600	MM_FRAME_FINE_TRANS	Fine offset for FRAME (SRAM)
18601	MM_NUM_GLOBAL_USER_FRAMES	Number of globally predefined user frames (SRAM)
18602	MM_NUM_GLOBAL_BASE_FRAMES	Number of global basic frames (SRAM)
18660	MM_NUM_SYNACT_GUD_REAL	Number of configurable real-type GUD variables
18661	MM_NUM_SYNACT_GUD_INT	Number of configurable integer-type GUD variables

Number	Identifier: $MN_	Description
18662	MM_NUM_SYNACT_GUD_BOOL	Number of configurable Boolean-type GUD variables
18663	MM_NUM_SYNACT_GUD_AXIS	Number of configurable axis-type GUD variables
18664	MM_NUM_SYNACT_GUD_CHAR	Configurable char-type GUD variable
18665	MM_NUM_SYNACT_GUD_STRING	Configurable STRING-type GUD variable
18700	MM_SIZEOF_LINKVAR_DATA	Size of the NCU link variable memory
18710	MM_NUM_AN_TIMER	Number of global time variables for synchronized actions
18720	MM_SERVO_FIFO_SIZE	Setpoint for buffer size between IPO and closed-loop position control
18780	MM_NCU_LINK_MASK	Activation of NCU link communication
18781	NCU_LINK_CONNECTIONS	Number of internal link connections
18782	MM_LINK_NUM_OF_MODULES	Number of NCU link modules
18790	MM_MAX_TRACE_LINK_POINTS	Size of trace data buffer for NCU link
18792	MM_TRACE_LINK_DATA_FUNCTION	Specifies the contents of NCU link files
18794	MM_TRACE_VDI_SIGNAL	Trace specification of VDI signals
18800	MM_EXTERN_LANGUAGE	Activation of external NC languages
18860	MM_MAINTENANCE_MON	Activate recording of maintenance data
18870	MM_MAXNUM_KIN_CHAINS	Maximum number of trains
18880	MM_MAXNUM_KIN_CHAIN_ELEM	Maximum number of train elements
18890	MM_MAXNUM_3D_PROT_AREAS	Maximum number of elements in 3D protection zones
18892	MM_MAXNUM_3D_PROT_AREA_ELEM	Maximum number of protection-zone elements
18894	MM_MAXNUM_3D_PROT_GROUPS	Maximum number of protection-zone groups
18896	MM_MAXNUM_3D_COLLISION	Maximum number of temporary memory locations for collision check

15.5.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20096	T_M_ADDRESS_EXIT_SPINO	Spindle number as address extension
27900	REORG_LOG_LIMIT	Percentage of IPO buffer for log-file enable
28000	MM_REORG_LOG_FILE_MEM	Memory size for REORG
28010	MM_NUM_REORG_LUD_MODULES	Number of modules for local user variables with REORG
28020	MM_NUM_LUD_NAMES_TOTAL	Number of local user variables
28040	MM_LUD_VALUES_MEM	Memory size for local user variables
28050	MM_NUM_R_PARAM	Number of channel-specific R parameters
28060	MM_IPO_BUFFER_SIZE	Number of NC blocks in the IPO buffer
28070	MM_NUM_BLOCKS_IN_PREP	Number of blocks for block preparation
28080	MM_NUM_USER_FRAMES	Number of settable frames

Number	Identifier: $MC_	Description
28081	MM_NUM_BASE_FRAMES	Number of basic frames (SRAM)
28082	MM_SYSTEM_FRAME_MASK	System frames (SRAM)
28083	MM_SYSTEM_DATAFRAME_MASK	System frames (SRAM)
28085	MM_LINK_TOA_UNIT	Allocation of a TO unit to a channel
28090	MM_NUM_CC_BLOCK_ELEMENTS	Number of block elements for compile cycles
28100	MM_NUM_CC_BLOCK_USER_MEM	Size of block memory for compile cycles
28105	MM_NUM_CC_HEAP_MEM	Heap memory in KB for compile cycle applications (DRAM)
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables
28160	MM_NUM_LINKVAR_ELEMENTS	Number of write elements for the NCU link variables
28180	MM_MAX_TRACE_DATAPOINTS	Size of trace data buffer
28200	MM_NUM_PROTECT_AREA_CHAN	Number of modules for channel-specific protection zones
28210	MM_NUM_PROTECT_AREA_ACTIVE	Number of simultaneously active protection zones
28212	MM_NUM_PROTECT_AREA_CONTOUR	Elements for active protection zones (DRAM)
28250	MM_NUM_SYNC_ELEMENTS	Number of elements for expressions in synchronized actions
28252	MM_NUM_FCTDEF_ELEMENTS	Number of FCTDEF elements
28254	MM_NUM_AC_PARAM	Dimension of $AC_PARAM.
28255	MM_BUFFERED_AC_PARAM	$AC_PARAM[ ] is saved in SRAM.
28256	MM_NUM_AC_MARKER	Dimension of $AC_MARKER
28257	MM_BUFFERED_AC_MARKER	$AC_MARKER[ ] is saved in SRAM.
28258	MM_NUM_AC_TIMER	Number of $AC_TIMER time variables (DRAM)
28274	MM_NUM_AC_SYSTEM_PARAM	Number of $AC_SYSTEM_ PARAM for motion-synchronous actions
28276	MM_NUM_AC_SYSTEM_MARKER	Number of $AC_SYSTEM_ MARKER for motion-synchronous actions
28290	MM_SHAPED_TOOLS_ENABLE	Enable tool-radius compensation for contour tools
28300	MM_PROTOC_USER_ACTIVE	Activate logging for a user
28301	MM_PROTOC_NUM_ETP_OEM_TYP	Number of ETP OEM event types
28302	MM_PROTOC_NUM_ETP_STD_TYP	Number of ETP standard event types
28400	MM_ABSBLOCK	Activate block display with absolute values
28402	MM_ABSBLOCK_BUFFER_CONF	Dimension size of upload buffer
28450	MM_TOOL_DATA_CHG_BUFF_SIZE	Buffer for tool data changes (DRAM)
28500	MM_PREP_TASK_STACK_SIZE	Stack size of preparatory task
28520	MM_MAX_AXISPOLY_PER_BLOCK	Maximum number of axis polynomials per block
28530	MM_PATH_VELO_SEGMENTS	Number of memory chips for limiting the tool-path velocity
28535	MM_FEED_PROFILE_SEGMENTS	Number of memory chips for feed profiles
28540	MM_ARCLENGTH_SEGMENTS	Number of memory chips for displaying the arc length function

Number	Identifier: $MC_	Description
28560	MM_SEARCH_RUN_RESTORE_MODE	Restore data after a simulation
28580	MM_ORIPATH_CONFIG	Setting for ORIPATH tool orientation trajectory referred to path

15.5.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
38000	MM_ENC_COMP_MAX_POINTS	Number of intermediate points with interpolatory compensation
38010	MM_QEC_MAX_POINTS	Number of values for quadrant-error compensation

T1: Indexing axes

16

16.1 Brief description

Indexing axes in machine tools

In certain applications, the axis is only required to approach specific grid points (e.g. location numbers). It is necessary to approach the defined grid points, the indexing positions, both in AUTOMATIC and set-up mode.

The relevant axes are called "indexing axes". The positions defined on the indexing axes are known as "coded positions" or "indexing positions".

Applications

Indexing axes are used predominantly in connection with specific types of tool magazines such as tool turrets, tool chain magazines or tool cartridge magazines. The coded positions refer to the individual locations of the tools in the magazines. During a tool change, the magazine is positioned at the location containing the tool to be loaded.

Display indexing

The following data can be queried via system variables:

- The number of the current indexing position:
  - When the "exact stop fine" window of the indexing position is reached
  - When half the distance to the next indexing position is crossed
- The programmed indexing position

16.2 Traversing of indexing axes

16.2.1 General information

Indexing axes can be traversed:

- Manually in the setting-up modes JOG and INC
- from one part program with special instructions for coded positions
- of PLC

Upon reaching the indexing position the following interface signal is given out to the PLC:

DB31, ... DBX76.6 (indexing axis in position)

Note

Hirth indexing axes cannot be traversed in JOG mode before reference point approach.

16.2.2 Traversing of indexing axes in the JOG mode

Reference point approach

An indexing axis approaches the reference point in the same way as other axes. The reference point does not have to coincide with an indexing position.

When reference point is reached:
DB31, ... DBX60.4 or 5 (referenced/synchronized 1 or 2) = 1
the indexing axis moves only to indexing positions in JOG mode during conventional and incremental traversing.

Exception: No indexing positions are approached when traversing with the handwheel.

If the axis is not referenced (DB31, ... DBX60.4 or 5 = 0), the indexing positions are ignored when traversing in JOG mode!

Continuous traversal in JOG

- Jog mode active:

  SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD = 1

  Pressing a "+" or "-" traversing key causes the indexing axis to move in the same way as with conventional JOG traversing. When the traversing key is released, the indexing axis traverses to the next indexing position in the direction of traversing.

- Continuous mode active:

  SD41040 $SN_JOG_CONT_MODE_LEVELTRIGGRD = 0

  Pressing the traversing key briefly (first rising signal edge) starts the traversing movement of the indexing axis in the desired direction. Traversing continues when the traversing key is released. When the traversing key is pressed again (second rising signal edge), the indexing axis traverses to the next indexing position in the direction of traversing.

Indexing axes are generally traversed in JOG mode (standard setting). Continuous mode plays a less important role.

If the operator changes the direction of traversing before the indexing position has been reached, the indexing axis is positioned on the next indexing position in the direction of traversing. The traversing movement must be started in the opposite direction.

For further information on continuous traversing in jog or continuous mode, please see:
References:
/FB2/ Function Manual, Extended Functions; Manual and Handwheel Travel (H1)

Incremental traversal in JOG mode (INC)

Irrespective of the current increment setting (INC1, ..., INCvar), the indexing axis always traverses **through one indexing position** in the selected direction when a traversing key "+" or "-" is pressed.

In jog mode, the traversing movement is interrupted when the traversing key is released. The indexing position can be approached by pressing the traversing key again.

In continuous mode, the traversing movement is interrupted when the traversing key is pressed again. The indexing axis is, in this case, not located on the indexing position.

Between indexing positions

If an indexing axis is situated between 2 indexing positions, then it approaches the next-higher indexing position when the "+" traversing key is pressed in JOG-INC mode. Similarly, pressing the "-" traversing key causes the next-lower indexing position to be approached.

Handwheel traversal

When the indexing axis is traversed by means of the handwheel in JOG mode, the **indexing positions are ignored**. As the handwheel is turned, the indexing axis moves to any position depending on the basic system in mm, inches or degrees.

If traversing of the indexing axis with the handwheel is to be interlocked, this can be handled by the PLC user program.

Signal from PLC "Indexing axis in position"

During the traversing motion of the indexing axis in the JOG mode, the following NC/PLC interface signal displays the reaching of the indexing position:

DB31, ... DBX76.6 (indexing axis in position)

Prerequisite: The indexing axis is referenced (DB31, ... DBX60.4 and 5 = 1)

Alarms in JOG mode

If the indexing axis leaves the traversing range defined in the indexing position table when traversing in JOG mode, alarm 20054 "wrong index for indexing axis in JOG" is output.

Rev. feedrate

In JOG mode, the response of the axis/spindle also depends on the setting data:

SD41100 $SN_JOG_REV_IS_ACTIVE (revolutional feed rate for JOG active)

SD41100	Description
= 1 (active)	The axis/spindle is always traversed with rrevolutional feed rate as a function of the master spindle: MD32050 $MA_JOG_REV_VELO (revolutional feed rate for JOG mode) or MD32040 $MA_JOG_REV_VELO_RAPID (revolutional feed rate for JOG with rapid traverse override)
= 0 (not active)	The response of the axis/spindle depends on the setting data: SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE(revolutional feed feed rate for position axes/spindles) The response of a geometry axis on which a frame acts is to rotate, depending on the setting data: SD42600 $SC_JOG_FEED_PER_REV_SOURCE

16.2.3 Traversing of indexing axes in the AUTOMATIC mode

Traversal to selected positions

An axis defined as an indexing axis can be made to approach **any selected position** from the NC part program in AUTOMATIC mode. This includes positions between the defined indexing positions.

These positions are programmed, in the usual way, in the unit of measurement (mm/inches or degrees) for the axis. The general programming instructions used for this purpose (`G90`, `G91`, `AC`, `IC`, etc.) are described in the Programming Manuals.

Traversal to "Coded positions"

Special instructions can also be used in the part program to traverse indexing axes to the "coded positions":

Statement	Effect
`CAC`	Approach absolute coded position
`CACP`	Approach absolute coded position in positive direction
`CACN`	Approach absolute coded position in negative direction
`CIC`	Approach incremental coded position
`CDC`	Approach coded position along direct (shortest) path

With absolute positioning, the indexing position to be approached is programmed, and with incremental positioning, the number of indexes to be traversed in the "+" or "-" direction is programmed.

On rotary axes, the indexing position can be approached directly across the shortest path (CDC) or with a defined direction of rotation (CACP, CACN).

Reaching the indexing position

If the "Exact stop fine" window is reached and the indexing axis is positioned on an indexing position, the following NC/PLC interface signal is enabled regardless of how the indexing position was reached.

DB31, ... DBX76.6 (indexing axis in position)

16.2.4 Traversing of indexing axes by PLC

Indexing axes can also be traversed from the PLC user program.

There are various methods:

- Concurrent positioning axes

  The indexing position to be approached can be specified by the PLC.

  References:
  /FB2/ Function Manual, Extended Functions; Positioning Axes (P2)

- Asynchronous subroutines (ASUBs)

  References:
  /FB1/ Function Manual, Basic Functions; Mode Group, Channel, Program Operation, Reset Response (K1)

16.3 Parameterization of indexing axes

Definition of the indexing axis

An axis (linear or rotary axis) can be defined as indexing axis with the axial machine data:

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB

Value	Description
0	The axis is not declared as an indexing axis.
1	The axis is an indexing axis. The associated indexing positions are stored in the indexing positions table 1.
2	The axis is an indexing axis. The associated indexing positions are stored in the indexing positions table 2.

Indexing position tables

The axis positions (in mm or degrees) assigned to the indexes must be stored for each indexing axis in the form of a table in machine data.

2 indexing position tables are possible:

MD10910 $MN_INDEX_AX_POS_TAB_1 [n] (indexing position table 1)

MD10930 $MN_INDEX_AX_POS_TAB_2 [n] (indexing position table 2)

Note

Several axes can be assigned to an indexing position table. On condition that these indexing axes are of the same type (linear axis, rotary axis, modulo 360° function).

No. of indexing positions

Up to 60 positions can be entered in each indexing position table:

[n = 0 ... 59]

The actually used number of entries is defined with the machine data:

MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1 ((number of positions of indexing position table 1)

MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2 ((number of positions of indexing position table 2)

Note

Entries in the indexing positions table that cross the parameterized number of indexing positions (MD10900 or MD10920) are not considered.

Valid measuring system

The indexing positions defined with MD10900 and MD10920 are related to the measuring system configured for position tables:

MD10270 $MN_POS_TAB_SCALING_SYSTEM

Value	Measuring system
0	Metric
1	inch

Note

MD10270 has an effect on the following setting data:

SD41500 $SN_SW_CAM_MINUS_POS_TAB_1 (switching point for falling cam edge 1-8)

to

SD41507 $SN_SW_CAM_PLUS_POS_TAB_4 (switching point for rising cam edge 25-32)

Entry for indexing positions

The following rules apply:

- The indexing positions must be entered in the table in ascending order (starting with the negative to the positive traversing range) with no gaps between the entries.
- Consecutive position values cannot be identical.
- The axis positions must be entered in the basic coordinate system.

Modulo rotary axis as indexing axis

The indexing axis is defined with Modulo 360° as rotary axis:

MD30300 $MA_IS_ROT_AX = 1

and

MD30310 $MA_ROT_IS_MODULO = 1

In this case, the following points must be observed additionally for the specification of the indexing positions:

- Permissible range: 0° ≤ Pos < 360°
- Since the indexing axis is defined as a continuously rotating rotary axis, indexing position 1 is approached after the highest valid indexing position in the table has been reached and the axis continues to traverse in the positive direction with INC. Similarly, indexing position 1 is followed by the highest valid indexing position in the negative direction with INC.

16.4 Programming of indexing axes

Coded position

To allow indexing axes to be positioned from the NC part program, special instructions are provided with which the indexing numbers (e.g. location numbers) are programmed instead of axis positions in mm or degrees. The availability of a special instruction depends on the axis type (linear or rotary axis):

Statement	Effect	Availability
CAC(i)	Traverse coded position in absolute terms	Linear axis, rotary axis
CACP(i)	Traverse coded position ain absolute terms in the positive direction	Rotary axis
CACN(i)	Traverse coded position ain absolute terms in the negative direction	Rotary axis
CDC(i)	Traverse coded position along the direct (shortest) path	Rotary axis
CIC(i)	Traverse coded position incrementally	Linear axis, rotary axis

i: Coded position (indexing position)

Value range of i: 0 ... 59; whole number (positive and negative values are possible in CIC)

Examples

Programming	Comment
POS[B]=CAC(20)	; Indexing axis B approaches the coded position (indexing) 20 in **absolute** mode. The direction of traversing depends on the current actual position.

Programming	Comment
POS[B]=CACP(10)	; Indexing axis B approaches the coded position (index position) 10 in **absolute mode with positive direction of rotation** (possible only for rotary axes).

Programming	Comment
POS[B]=CACN(10)	; Indexing axis B approaches the coded position (index position) 10 in **absolute mode with negative direction of rotation** (possible only for rotary axes).

Programming	Comment
POS[B]=CDC(50)	; Indexing axis B approaches indexing position 50 **directly along the shortest path** (possible only for rotary axes).

Programming	Comment
POS[B]=CIC(-4)	; Indexing axis B traverses four indexing positions **incrementally** from its current position. **in a negative direction.**

Programming	Comment
POS[B]=CIC(35)	; Indexing axis B traverses 35 indexing positions **incrementally** from its present position **in a positive direction.**

Special features

- Modulo rotary axis as indexing axis

  On modulo rotary axes, the indexing positions are divided in factors of 360° and approached directly.

- Indexing axis is between two indexing positions

  The specified position instructions have the following effect in the AUTOMATIC mode.

POS[B]=CIC(1)	The **next higher** indexing position is approached.
POS[B]=CIC(-1)	The **next lower** indexing position is approached.
POS[B]=CIC(0)	The indexing axis is **not** traversed.

Display of indexing position

The number of the indexing position programmed last can be read with the following system variables:

$AA_PROG_INDEX_AX_POS_NO

The number of the indexing position traversed last can be displayed with the following system variables:

$AA_ACT_INDEX_AX_POS_NO

The display depends on the setting in machine data:

MD10940 $MN_INDEX_AX_MODE (settings for indexing position)

Bit	Value	Description
0	0	The indexing position changes when the indexing position is reached ("exact stop fine" window) and remains unchanged until the next indexing position is reached. The indexing area thus begins at one indexing position and ends in front of the next indexing position.
	1	The indexing position changes when half the indexing position is reached. A quasi-symmetrical indexing area is thus applied around the indexing position (symmetrical only on linear axes with equidistant indexing or modulo rotary axes on which the indexing area is an integer multiple of the modulo range (MD30330 $MA_MODULO_RANGE), otherwise proportional to the distances between the indexing positions). On **modulo rotary axes**, the area between the last indexing position and the first indexing position is **divided proportionally** based on the lengths of the first indexing area and the last indexing area.

The following graphics will illustrate the difference between Bit 0 = 0 and Bit 0 = 1:

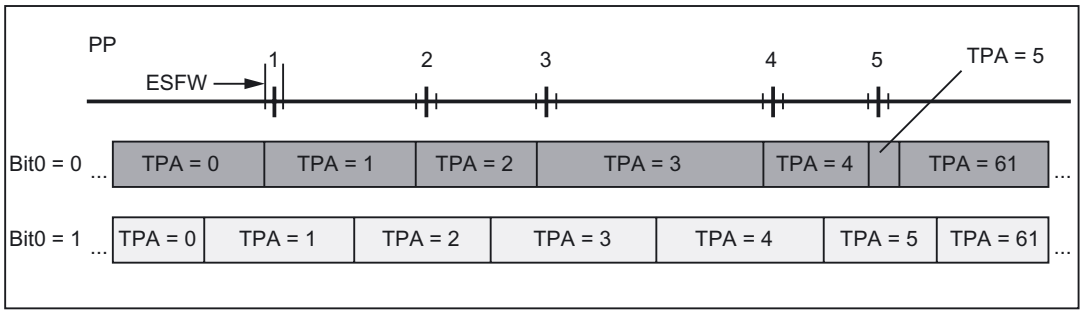
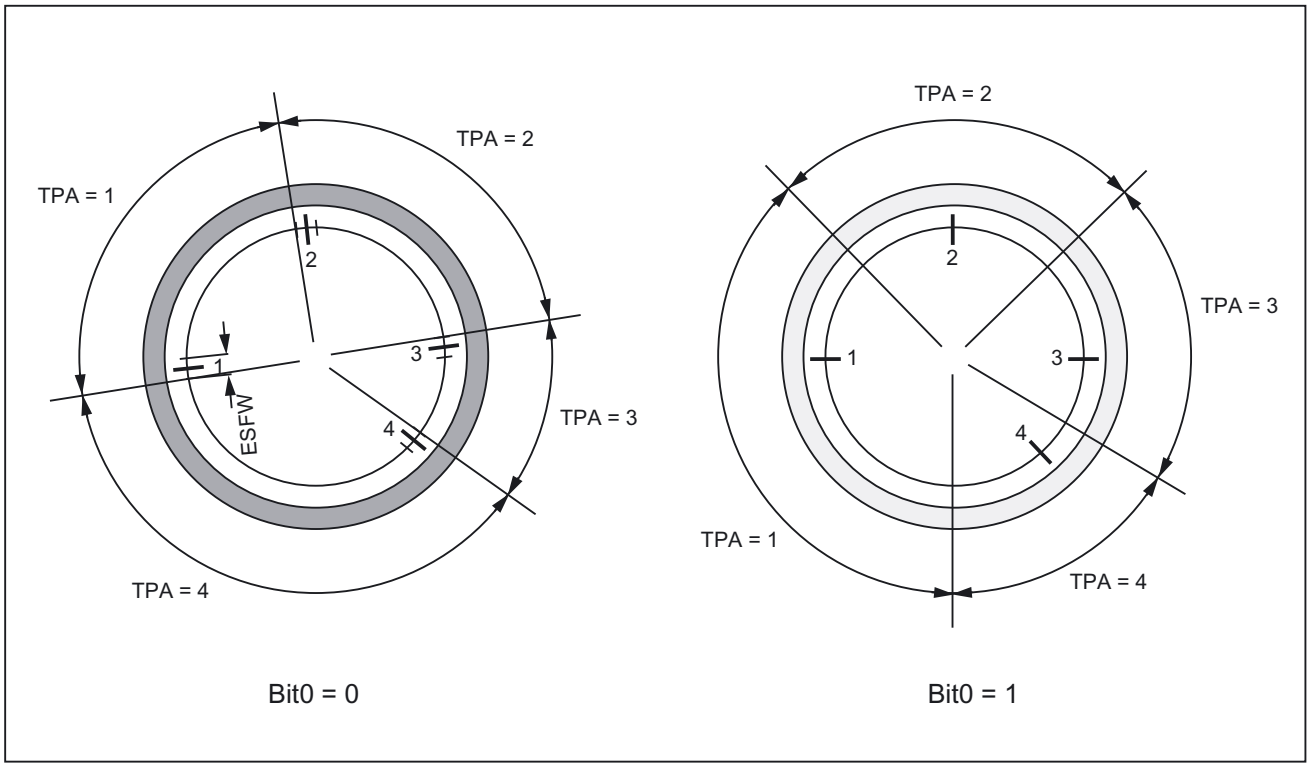


Figure 16-1 Indexing position displays: Linear axis



TP Programmed indexing position
TPA Displayed indexing position
ESFW "Exact stop fine" window

Figure 16-2 Indexing position displays: Modulo rotary axis

Value range of $AA_ACT_INDEX_AX_POS_NO

Expected value ranges of system variables $AA_ACT_INDEX_AX_POS_NO:

Indexing positions from table			
Modulo rotary axis	1 ... n	None 0 n = maximum 60	
Linear axis	0*, 1, 2, 3, ... 59, 60, 61*	0*:	**below** total indexing area
		61*:	**above** total indexing area

Equidistant indexing positions		
Modulo rotary axis	1 ... m	None 0 m = denominator (counter)
Linear axis	... -3, -2, -1, 0, 1, 2, 3, ...	

Traversing to the next indexing position

The response to the "Travel to the next indexing position" command depends on the setting in machine data:

MD10940 $MN_INDEX_AX_MODE (settings for indexing position)

Bit	Value	Description
0	0	The next indexing position is approached.
	1	The next indexing position in the direction of motion is always approached.

The following example will serve as explanation:

Bit 0 = 1 and axis below indexing position (but outside "exact stop fine" window).

Although the system variable $AA_ACT_INDEX_AX_POS_NO is indicating indexing position 2, indexing position 2 and **not** indexing position 3 is approached with the "Traverse to next position" command. The next indexing position (in this case indexing position 3) is not approached with the "Traverse to next position" command until the axis is located exactly at (exact stop fine) or above the indexing position.

The nearest indexing position in the current direction of motion is always approached! Under certain circumstances, it is thus necessary to transmit the "traverse to next position" command twice to move from the currently displayed indexing position to the next indexing position number (e.g. from 2 to 3).

FRAMES

Since the control interprets the positions stored in the indexing position table in mm, inches or degrees, FRAMES are not interlocked on indexing axes.

FRAMES are not usually needed for indexing axes, depending on the area of application. It is advisable in most cases to suppress FRAMES and work offsets for indexing axes in the part program.

16.5 Equidistant index intervals

16.5.1 Function

General information

The following exist:

- Any number of equidistant index intervals
- Modified action of MD for indexing axes

Equidistant index intervals can be used for:

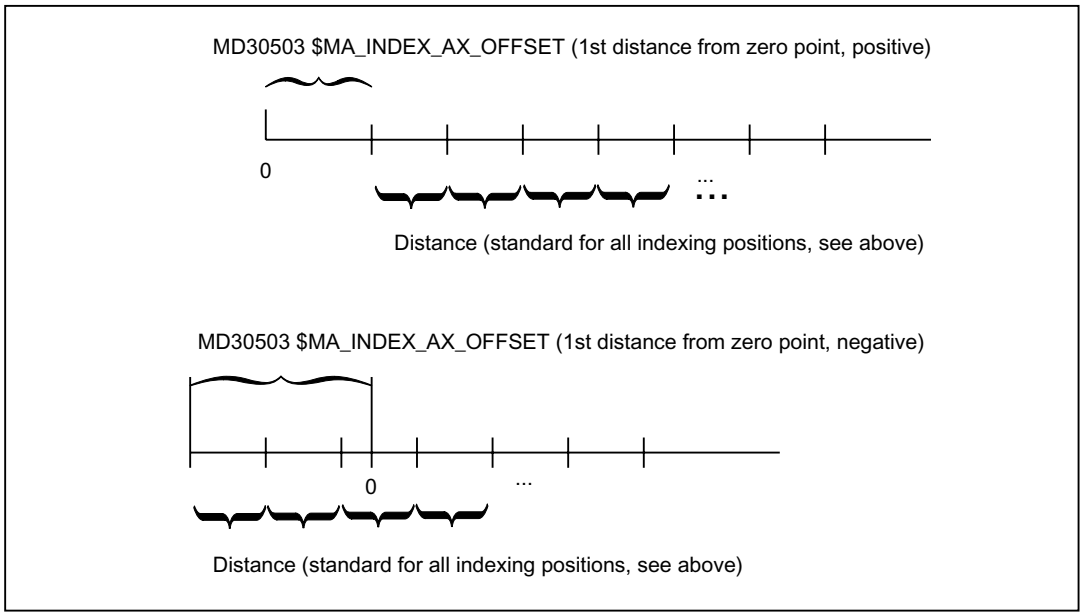
- Linear axes
- Modulo rotary axes
- Rotary axes

Distance between indexes

The index distance is determined for equidistant index intervals according to the following formula:

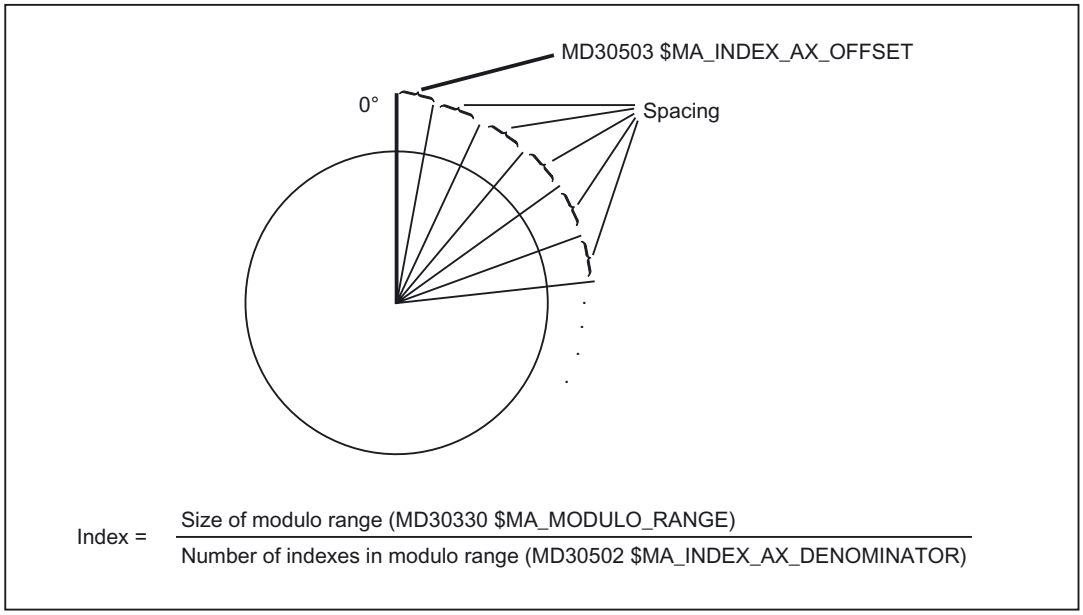
$$\text{Distance} = \frac{\text{Numerator (MD30501 \$MA_INDEX_AX_NUMERATOR)}}{\text{Denominator (MD30502 \$MA_INDEX_AX_DENOMINATOR)}}$$

Linear axis



Modulo rotary axis

$$\text{Index} = \frac{\text{Numerator (MD30330 \$MA_MODULO_RANGE)}}{\text{Denominator (MD30502 \$MA_INDEX_AX_DENOMINATOR)}}$$



Activating

The functions with equi-distant indexing for an axis (linear axis, modulo rotary axis or rotary axis) is activated in the following settings

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[axis] = 3

16.5.2 Hirth tooth system

Function

With Hirth tooth systems, positions of rotation on a rotary axis are usually interlocked using a latch or other toothed wheel via a linear axis. The interlock should only be activated when an indexing position has been reached precisely. The distance between the indexing positions is the same (equidistant) across the entire circumference.

Requirements

The rotary axis must be an indexing axis. The axis must be referenced.

References:
/FB1/ Function Manual, Basic Functions; Reference Point Approach (R1)

Activation

The following machine data must be set to 1:
MD30505 $MA_HIRTH_IS_ACTIVE

The machine data must be set to 3 (equidistant indexing positions):
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB

Activation

- The rotary axis can only approach indexing positions in all modes and operating states.
- In JOG mode, the axis can be traversed under JOG control or incrementally.

  Prerequisites: The axis is referenced.
- Jogging with the handwheel is not possible.

  References:
  /FB2/ Function Manual, Extended Functions; Jog With/Without Handwheel (H1)
- Only "coded positions" can be approached in AUTO, MDA or via ASUBs.
- The PLC can only move the axis to indexing positions.

16.5.3 Response of the Hirth axes in particular situations

STOP/RESET

When NC-STOP and RESET are performed during a traversing movement, the next indexing position is approached.

EMERGENCY STOP

After an EMERGENCY STOP, the PLC or the operator must move the indexing axis back to an indexing position in JOG mode before the longitudinal axis can be moved in/down.

Override = 0 or signal "axis stopped"

If the axis has already moved away from the previous indexing position when these events occur, the control moves the axis to the next possible indexing position before the response is initiated.

Deletion of distance-to-go

After traversing to the next possible indexing position, the movement is aborted at this position.

Command axes

If MOV = 0 is specified for a moving command axis, the axis continues traversing to the next possible indexing position.

References:
/FBSY/ Function Manual, Synchronized Actions

MOV command

`MOV = 1`	Works on indexing axes with and without Hirth tooth system.
`MOV = 0`	Same function for both: approaches the next position.

DELDTG command

In the case of indexing axes without Hirth tooth system:	Axis stops immediately.
In the case of indexing axes with Hirth tooth system:	Axis traverses to next position.

16.5.4 Restrictions

Transformations

The axis for which the Hirth tooth system is defined cannot take part in kinematic transformations.

PRESET

The axis for which the Hirth tooth system is defined cannot be set to a new value with `PRESET`.

Rev. feedrate

The axis for which the Hirth tooth system is defined cannot be traversed at revolutional feedrate.

Path/velocity overlay

The axis for which the Hirth tooth system is defined cannot be used with path or velocity overlay.

Frames, ext. work offset, DRF

The axis for which the Hirth tooth system is defined does not support frames or interpolation compensation such as external work offsets, DRF, etc.

Couplings

A Hirth tooth system axis can never be one of the following axis types:

- following axis with master value coupling
- coupled-motion axis
- gantry following axis

References:
/FB3/ Function Manual, Special Functions; Coupled axes and ESR (M3)

16.5.5 Modified activation of machine data

RESET

A RESET is required in order to activate the following machine data after new values have been assigned to them.

MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1

MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2

MD10910 $MN_INDEX_AX_POS_TAB_1

MD10930 $MN_INDEX_AX_POS_TAB_2

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB

16.6 Starting up indexing axes

Procedure

The procedure for starting up indexing axes is identical to normal NC axes (linear and rotary axes).

Rotary axis

If the indexing axis is defined as a rotary axis (MD30300 $MA_IS_ROT_AX = "1") with modulo 360° conversion (MD30310 $MA_ROT_IS_MODULO = "1"), indexing positions are also approached with modulo 360°. Only positions within the range from 0° to 359.999° can then be entered in the indexing position table. Otherwise alarm 4080 "Incorrect configuration for indexing axis in MD [Name]" is output during power-up.

The position display can be set to modulo 360° with machine data:
MD30320 $MA_DISPLAY_IS_MODULO = 1.

Special machine data

The following machine data must be set in addition:

General machine data	
MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1	Number of positions for indexing axis table 1
MD10920 $MN_INDEX_AX_LENGTH_POS_TAB_2	Number of positions for indexing axis table 2
MD10910 $MN_INDEX_AX_POS_TAB_1 [n]	Indexing position table 1
MD10930 $MN_INDEX_AX_POS_TAB_2 [n]	Indexing position table 2

Axial machine data	
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB	Axis is indexing axis (assignment of indexing position table 1 or 2, or 3 for equidistant indexing)
MD30505 $MA_HIRTH_IS_ACTIVE	Axis has "Hirth tooth system" property
MD30501 INDEX_AX_NUMERATOR	Numerator for equidistant indexing
MD30502 INDEX_AX_DENOMINATOR	Denominator for equidistant indexing
MD30503 INDEX_AX_OFFSET	Distance of 1st indexing position from zero

Machine data examples

The assignment of the above machine data is described in the following paragraphs using two examples.

Example of indexing axis as rotary axis

Tool turret with 8 locations. The tool turret is defined as a continuously rotating rotary axis. The distances between the 8 turret locations are constant. The first turret location is located at position 0°:

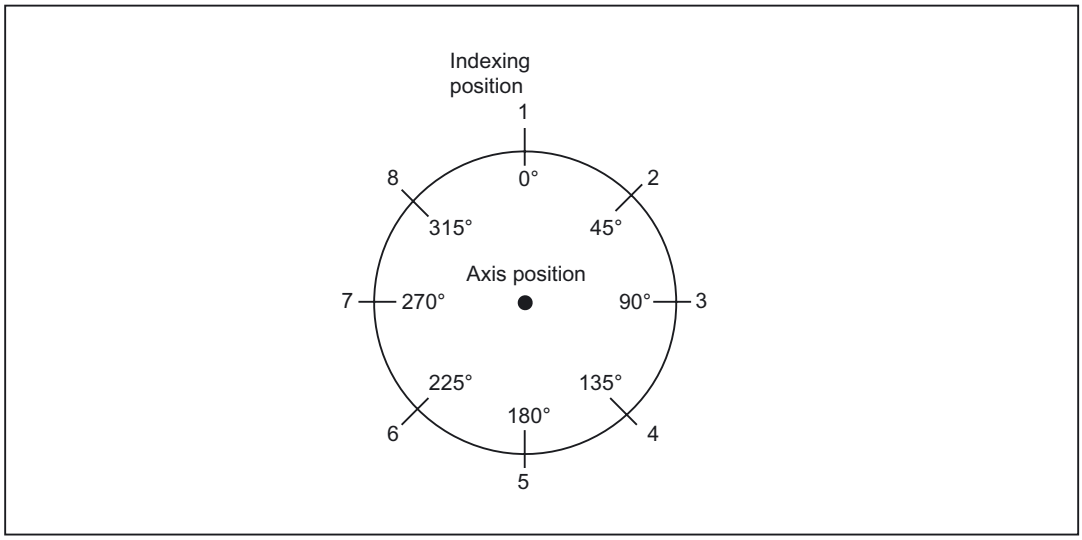


Figure 16-3 Example: Tool turret with 8 locations

The indexing positions for the tool turret are entered in indexing position table 1.

```
MD10910 $MN_INDEX_AX_POS_TAB_1[0] = 0       ;  1st indexing position at 0°
MD10910 $MN_INDEX_AX_POS_TAB_1[1] = 45      ;  2nd Indexing position at 45°
MD10910 $MN_INDEX_AX_POS_TAB_1[2] = 90      ;  3rd Indexing position at 90°
MD10910 $MN_INDEX_AX_POS_TAB_1[3] = 135     ;  4th Indexing position at 135°
MD10910 $MN_INDEX_AX_POS_TAB_1[4] = 180     ;  5th Indexing position at 180°
MD10910 $MN_INDEX_AX_POS_TAB_1[5] = 225     ;  6th Indexing position at 225°
MD10910 $MN_INDEX_AX_POS_TAB_1[6] = 270     ;  7th Indexing position at 270°
MD10910 $MN_INDEX_AX_POS_TAB_1[7] = 315     ;  8th Indexing position at 315°
```

Other machine data:

```
MD10900 $MN_INDEX_AX_LENGTH_POS_TAB_1= 8    ;  8 indexing positions in table 1
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB [AX5]   ;  Axis 5 is defined as indexing
= 1                                         ;  axis
                                               Indexing positions in table 1
MD30300 $MA_IS_ROT_AX [AX5] = 1             ;  Axis 5 is rotary axis
MD30310 $MA_ ROT_IS_MODULO [AX5] = 1        ;  Modulo conversion is activated
```

Example of indexing axis as linear axis

Workholder with 10 locations.

The distances between the 10 locations are different. The first location is at position -100 mm.

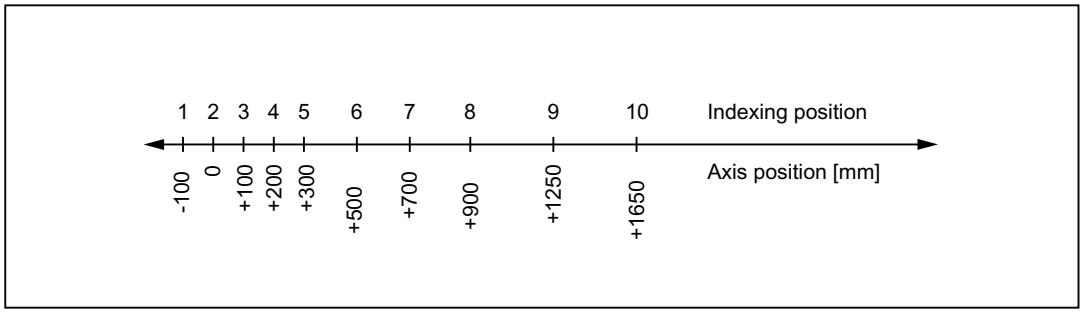


Figure 16-4 Example: Workholder as an indexing axis

The indexing positions for the workholder are entered in table 2:

```
MD10930 $MN_INDEX_AX_POS_TAB_2[0] = -100    ;  1st indexing position at -100
MD10930 $MN_INDEX_AX_POS_TAB_2[1] = 0       ;  2nd indexing position at 0
MD10930 $MN_INDEX_AX_POS_TAB_2[2] = 100     ;  3rd indexing position at 100
MD10930 $MN_INDEX_AX_POS_TAB_2[3] = 200     ;  4th indexing position at 200
MD10930 $MN_INDEX_AX_POS_TAB_2[4] = 300     ;  5th indexing position at 300
MD10930 $MN_INDEX_AX_POS_TAB_2[5] = 500     ;  6th indexing position at 500
MD10930 $MN_INDEX_AX_POS_TAB_2[6] = 700     ;  7th indexing position at 700
MD10930 $MN_INDEX_AX_POS_TAB_2[7] = 900     ;  8th indexing position at 900
MD10930 $MN_INDEX_AX_POS_TAB_2[8] = 1250    ;  9th indexing position at 1250
MD10930 $MN_INDEX_AX_POS_TAB_2[9] = 1650    ;  10th indexing position at 1650
```

Other machine data

```
MD10920                                     ; 10 indexing positions in table 2
$MN_INDEX_AX_LENGTH_POS_TAB_2=10
MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB [AX6]   ; Axis 6 is defined as indexing
= 2                                         ; axis
                                              Indexing positions in table 2
```

16.7 Special features of indexing axes

DRF

An additional incremental work offset can also be generated for indexing axes in AUTOMATIC mode with the handwheel using the DRF function.

Software limit switch

The software limit switches are also effective during traversing movements once the indexing axis has been referenced.

In handwheel traversing with JOG continuous or JOG incremental, the indexing axis stops at the indexing position ahead of the software limit switch.

Reference point approach

When **reference point is reached:**
DB31, ... DBX60.4 or 60.5 (referenced/synchronized 1 or 2) = 1
the indexing axis moves only to indexing positions in JOG continuous and JOG incremental mode.

If the axis is not referenced:
DB31, ... DBX60.4 or 60.5 (referenced/synchronized 1 or 2) = 0,
the indexing positions are ignored when traversing in JOG mode!

Since the axis positions stored in the indexing position tables only correspond to the machine positions when the axis is referenced, an NC start must be disabled for as long as the indexing axis is not referenced.

Position display

Positions on indexing axes are displayed in the units of measurement normally used for the axes (mm, inches or degrees).

Abort through RESET

RESET causes the traversing movement on an indexing axis to be aborted and the axis to be stopped. The indexing axis is no longer positioned on an indexing position.

16.8 Examples

16.8.1 Examples of equidistant indexes

Modulo rotary axis

MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] =18

MD30503 $MA_INDEX_AX_OFFSET[AX4]=5

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3

MD30300 $MA_IS_ROT_AX[AX4] = TRUE

MD30310 $MA_ROT_IS_MODULO[AX4] = TRUE

With the machine data above, axis 4 is defined as a modulo rotary axis and an indexing axis with equidistant positions every 20° starting at 5°.
The following indexing positions result:
5, 25, 45, 65, 85, 105, 125, 145, 165, 185, 205, 225, 245, 265, 285, 305, 325 and 245 degrees.

Note

The assignment:
MD30502 $MA__INDEX_AX_DENOMINATOR[AX4] =18
produces a 20° division because the default for machine data:
MD30330 $MA_MODULO_RANGE
is 360°.

Rotary axis

MD30501 $MA_INDEX_AX_NUMERATOR[AX4] = 360

MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] =18

MD30503 $MA_INDEX_AX_OFFSET[AX4]=100

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3

MD30300 $MA_IS_ROT_AX[AX4] = TRUE

MD36100 $MA_POS_LIMIT_MINUS[AX1]=100

MD36110 $MA_POS_LIMIT_PLUS[AX1]=260

With the machine data above, axis 4 is defined as a rotary axis and an indexing axis with equidistant positions every 20° starting at 100°.
The following indexing positions result:
100°, 120°, 140° etc.
Positions less than 100° cannot be approached as indexing positions.
It is advisable to place the lower software limit switch in this case. The indexing positions continue until the software limit switch is reached (in this case 260°). The rotary axis can therefore only traverse between 100° and 260°.

Linear axis

MD30501 $MA_INDEX_AX_NUMERATOR[AX1] = 10

MD30502 $MA_INDEX_AX_DENOMINATOR[AX1] =1

MD30503 $MA_INDEX_AX_OFFSET[AX1]=-200

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX1] = 3

MD30300 $MA_IS_ROT_AX[AX1] = FALSE

MD36100 $MA_POS_LIMIT_MINUS[AX1]=-200

MD36110 $MA_POS_LIMIT_PLUS[AX1]=200

With the machine data above, axis 4 is defined as a linear axis and an indexing axis with equidistant positions every 10 mm starting at -200 mm.
The following indexing positions result:
-200, -190, -180 mm etc.
These indexing positions continue until the software limit switch is reached (in this case 200 mm).

Hirth tooth system

MD30502 $MA_INDEX_AX_DENOMINATOR[AX4] =360

MD30503 $MA_INDEX_AX_OFFSET[AX4]=0

MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB[AX4] = 3

MD30300 $MA_IS_ROT_AX[AX4] = TRUE

MD30310 $MA_ROT_IS_MODULO[AX5] = TRUE

MD30505 $MA_HIRTH_IS_ACTIVE[AX4] = TRUE

With the machine data above, axis 4 is defined as a modulo rotary axis and an indexing axis with Hirth tooth system and equidistant positions every 1° starting at 0°.

16.9 Data lists

16.9.1 Machine data

16.9.1.1 General machine data

Number	Identifier: $MN_	Description
10260	CONVERT_SCALING_SYSTEM	Basic system switchover active
10270	POS_TAB_SCALING_SYSTEM	System of measurement of position tables
10900	INDEX_AX_LENGTH_POS_TAB_1	Number of positions for indexing axis table 1
10910	INDEX_AX_POS_TAB_1[n]	Indexing position table 1
10920	INDEX_AX_LENGTH_POS_TAB_2	Number of positions for indexing axis table 2
10930	INDEX_AX_POS_TAB_2[n]	Indexing position table 2
10940	INDEX_AX_MODE	Options for indexing positions

16.9.1.2 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
30300	IS_ROT_AX	Rotary axis
30310	ROT_IS_MODULO	Modulo conversion for rotary axis
30320	DISPLAY_IS_MODULO	Position display is modulo 360°
30500	INDEX_AX_ASSIGN_POS_TAB	Axis is indexing axis
30501	INDEX_AX_NUMERATOR	Numerator for indexing axes with equidistant positions
30502	INDEX_AX_DENOMINATOR	Denominator for indexing axes with equidistant positions
30503	INDEX_AX_OFFSET	Indexing position for indexing axes with equidistant positions
30505	HIRTH_IS_ACTIVE	Hirth tooth system is active

16.9.2 Setting data

16.9.2.1 General setting data

Number	Identifier: $SN_	Description
41050	JOG_CONT_MODE_LEVELTRIGGRD	JOG continuous in inching mode

16.9.3 Signals

16.9.3.1 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Referenced/synchronized 1, referenced/synchronized 2	DB31, ...DBX60.4/5	DB390x.DBX0.4/5
Indexing axis in position	DB31, ...DBX76.6	DB390x.DBX1002.6

16.9.4 System variables

16.9.4.1 System variables

Identifier	Description
$AA_ACT_INDEX_AX_POS_NO[axis]	Number of last indexing position reached or overtraveled
$AA_PROG_INDEX_AX_POS_NO[axis]	Number of programmed indexing position

W3: Tool change

17

17.1 Brief description

Tool change

CNC-controlled machine tools are equipped with tool magazines and automatic tool change facility for the complete machining of workpieces.

Sequence

The procedure for changing tools comprises three steps:

1. Movement of the tool carrier from the machining position to the tool change position
2. Tool change
3. Movement of the tool carrier from the tool change position to the new machining position.

Requirements

The following is required for tool change:

- Short idle times
- Time-saving searches, provision and return of tool during the machining time.
- Simple programming of the tool change cycle
- Automatic flow of the required axis and gripper movements
- Easy fault recovery

17.2 Tool magazines and tool change equipments

Tool magazines and tool changing equipment are selected according to the machine type:

Machine type	Tool magazine	Tool change equipment
Turning machines	Turret (disk, flat, inclined)	No special tool change equipment. The tool is changed by turning the turret
Milling machines	Magazines (chain, disk-type, rotary-plate, cartridge)	Gripper/dual gripper as tool change equipment.

As the changing operation interrupts the machining, idle times times must be minimized.

17.3 Tool change times

Tool change times depend strongly on the design layout of the machine tool.

Typical tool change times	
0.1 to 0.2 s	for advancing a turret
0.3 to 2 s	for tool change with gripper for a prepared tool

17.4 Cut-to-cut time

The cut-to-cut time is the period that elapses when a tool is changed between retraction from the interruption point on the contour (from cut) and repositioning at the interruption point (return to cut) with the new tool when the spindle is rotating.

Typical cut-to-cut times are as follows:

Typical cut-to-cut times	
0.3 to 1 s	for turning machine with turret
0.5 to 5 s	for milling machine with automatic tool changer

17.5 Starting the tool change

Variants

The tool change can be actuated by:

- T function
- M command (preferably M06).

Parameter setting

Which control versions should be effective is defined with the machine data:

MD22550 $MC_TOOL_CHANGE_MODE

Value	Description	Typical application
0	The T function loads the new tool immediately.	Turning machines with tool turret
1	New tool with T function prepared for change and placed in the tool change position simultaneously during the machining time. The M command is used to remove the old tool from the spindle and load the new tool.	Milling machines with a tool magazine,

The M command for tool change is defined in machine data:

MD22560 $MC_TOOL_CHANGE_M_CODE

Default setting is 6 (corresponding to DIN 66025).

Note

If the tool offset number is supplied from the PLC or an HMI tool manager, a block search stop STOPRE must be inserted at a suitable point. STOPRE must be avoided, however, when tool radius compensation (G41 / G42) or SPLINE interpolation are active, since several blocks are required here in advance for the path calculation.

References

You will find further information about M functions which also apply to tool change M06 (e.g. extended address, time of output to PLC, auxiliary function groups, behavior during block search, behavior during overstore) in:

/FBSY/ Function Manual, Synchronized Actions

17.6 Tool change point

Tool change point

The selection of the tool change point has a significant effect on the cut-to-cut time (Page 968). The tool change point is chosen according to the machine tool concept and, in certain cases, according to the current machining task.

Fixed point approach

Fixed positions on a machine axis stored in machine data can be approached by means of the fixed-point approach function. This can be used to define and control one or several tool change points.

Fixed point approach is possible in two ways:

- Fixed point approach in JOG

  The machine user starts the fixed point approach in the JOG mode with the traverse keys or the handwheel.

  References:
  Function Manual Extended Functions; Handwheel and handwheel frame (H1), Chapter: "Fixed point approach in JOG"

- Fixed point approach with `G75`

  Fixed point approach is called from the part program via the command `G75`.

  References:
  Programming Manual Fundamentals, Chapter: "Travel commands" > "Approaching fixed point (G75)"

17.7 Supplementary conditions

The tool change requires, amongst other things, a tool management system which ensures that the tool to be loaded is available at the tool change position at the right time.

17.8 Examples

Milling machine

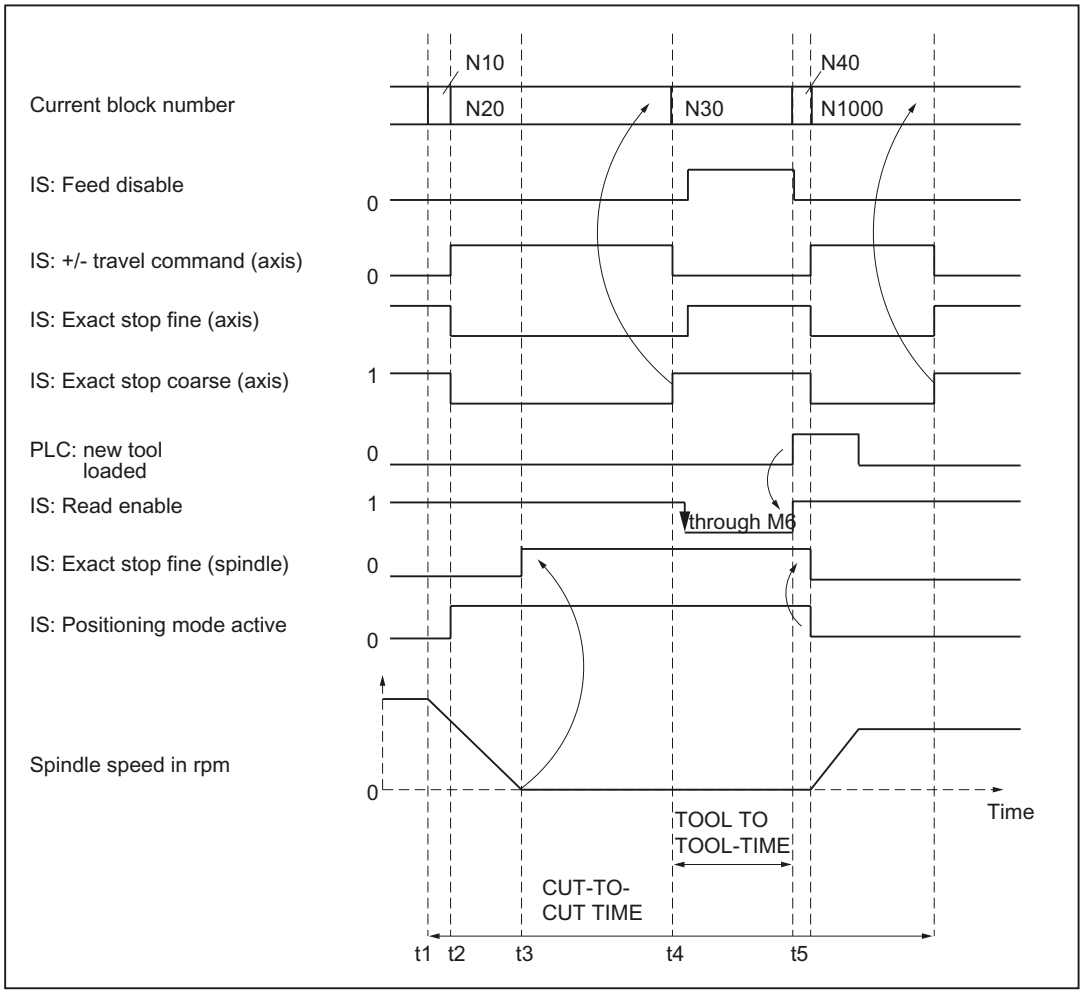
The following example shows a typical cut-to-cut sequence of operations for a tool change with a tool changer and a fixed absolute tool change point on a milling machine.

Machining program:

```
N970 G0 X= Y= Z= LF                          ;  Retraction from contour
N980 T1 LF                                   ;  Tool preselection
N990 W_WECHSEL LF                            ;  Subroutine call without parameters
N1000 G90 G0 X= Y= Z= M3 S1000 LF            ;  Continue machining
```

Subroutine for tool change:

```
PROC W_WECHSEL LF
N10 SPOSA= S0 LF                             ;  Spindle positioning
N20 G75 FP=2 X1=0 Y1=0 Z1=0                  ;  Approach tool change position
N30 M06 LF                                   ;  Change tool
N40 M17 LF
```



t_1: Axes stationary.
Spindle rotates.
Start of tool change cycle in `N10`.

t_2: Move axes to tool change point with `G75` in `N20`.

t_3: Spindle reaches programmed position from block `N10`.

t_4: Axes reach exact stop coarse from `N20`; `N30` thus begins:
`M06` removes the previous tool from the spindle and loads and clamps the new tool.

t_5: Tool changer swivels back to original position.

Figure 17-1 Chronological sequence of tool change

Then, in `N1000` of the calling main program:

- The new tool offset can be selected
- the axes can be returned to the contour, or
- the spindle can be accelerated.

17.9 Data lists

17.9.1 Machine data

17.9.1.1 General machine data

Number	Identifier: $MN_	Description
18082	MM_NUM_TOOL	Number of tools

17.9.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
22200	AUXFU_M_SYNC_TYPE	Output timing of M functions
22220	AUXFU_T_SYNC_TYPE	Output timing of T functions
22550	TOOL_CHANGE_MODE	New tool offset for M function
22560	TOOL_CHANGE_M_CODE	M function for tool change

17.9.1.3 Axis-/spindle-specific machine data

Number	Identifier: $MA_	Description
30600	FIX_POINT_POS[n].	Fixed point positions of the machine axes with G75

17.9.2 Signals

17.9.2.1 Signals from channel

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
M function M06	DB21, ...DBX194.6	DB2500.DBB1000.6

W4: Grinding-specific tool offset and monitoring functions - only 840D sl

18

18.1 Brief description

Contents

The topics of this functional description are:

- Grinding-specific tool offset
- Online tool offsets (continuous dressing)
- Grinding-specific tool monitoring
- Constant grinding wheel peripheral speed (GWPS)

References

For fundamentals see:
/FB1/ Function Manual Basic Functions; Tool Offset (W1)

For information about programming, mode of operation and handling, please refer to
/PG/ Programming Manual Fundamentals

18.2 Tool offset for grinding operations

18.2.1 Structure of tool data

Grinding tools

Grinding tools are tools of types 400 to 499.

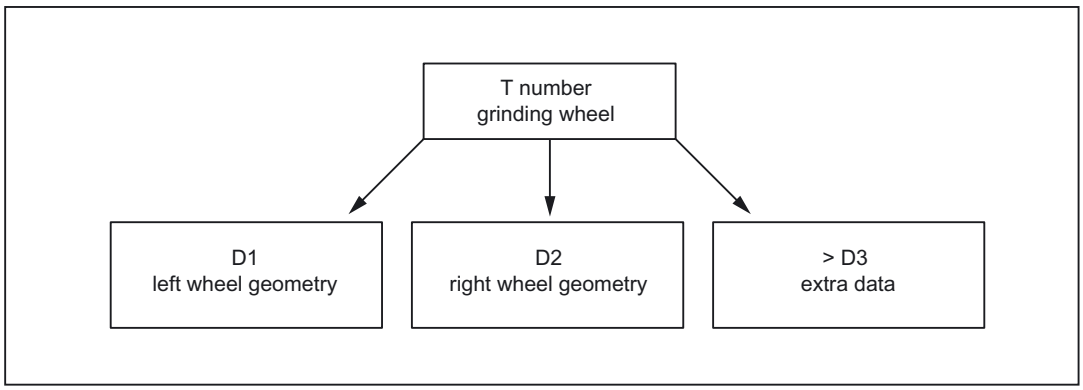
Tool offset for grinding tools

Grinding tools normally have specific tool and dresser data in addition to cutting edge data.

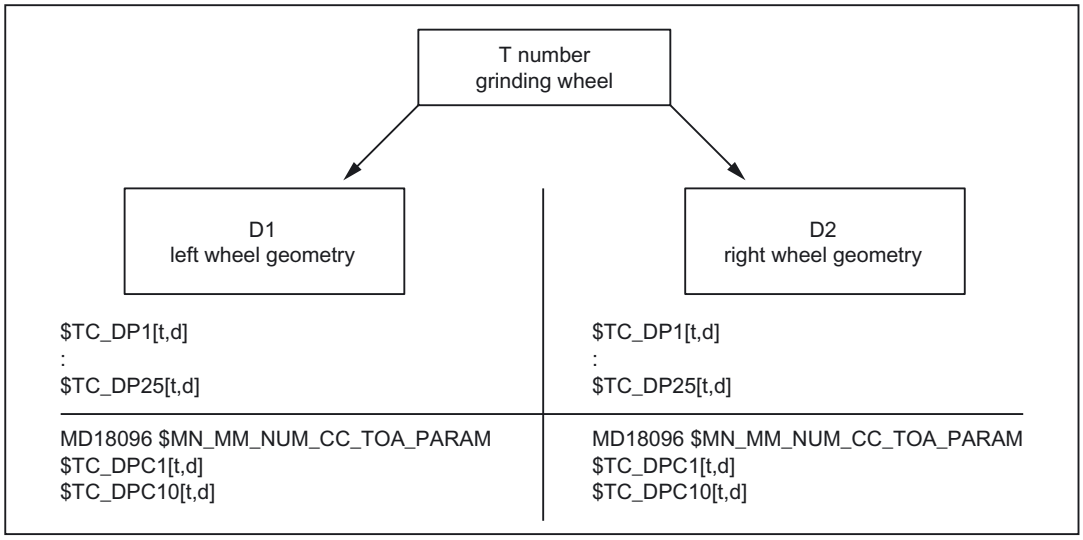
The specific grinding wheel data for the left and right wheel geometry can be stored under a T number in `D1` and `D2`.

If data are needed for the dresser geometry, they can be stored, e.g., starting at `D3` of a T number or in additional cutting-edge-specific data (MD18096 $MN_MM_NUM_CC_TOA_PARAM).

Example 1:



Example 2:



All offsets belonging to a grinding wheel and dresser can be combined in the tool edges `D1` and `D2` for the grinding wheel and, for example, `D3` and `D4` for the dresser:

- `D1`: grinding wheel geometry left
- `D2`: Grinding wheel geometry right
- `D3`: Dresser geometry left
- `D4`: Dresser geometry right

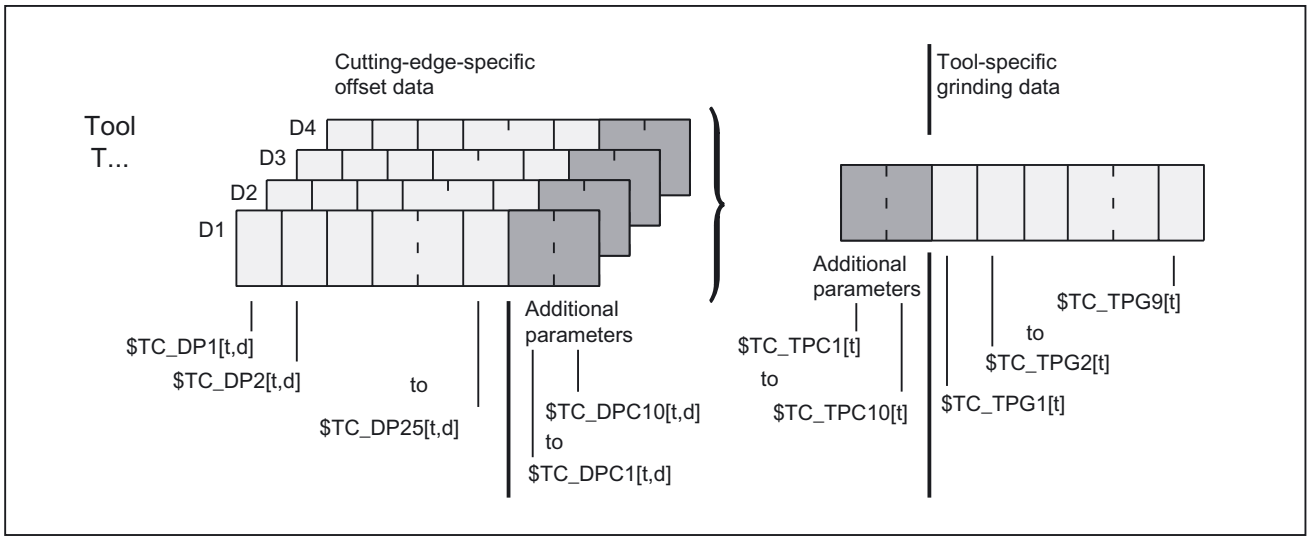


Figure 18-1 Structure of tool offset data for grinding tools

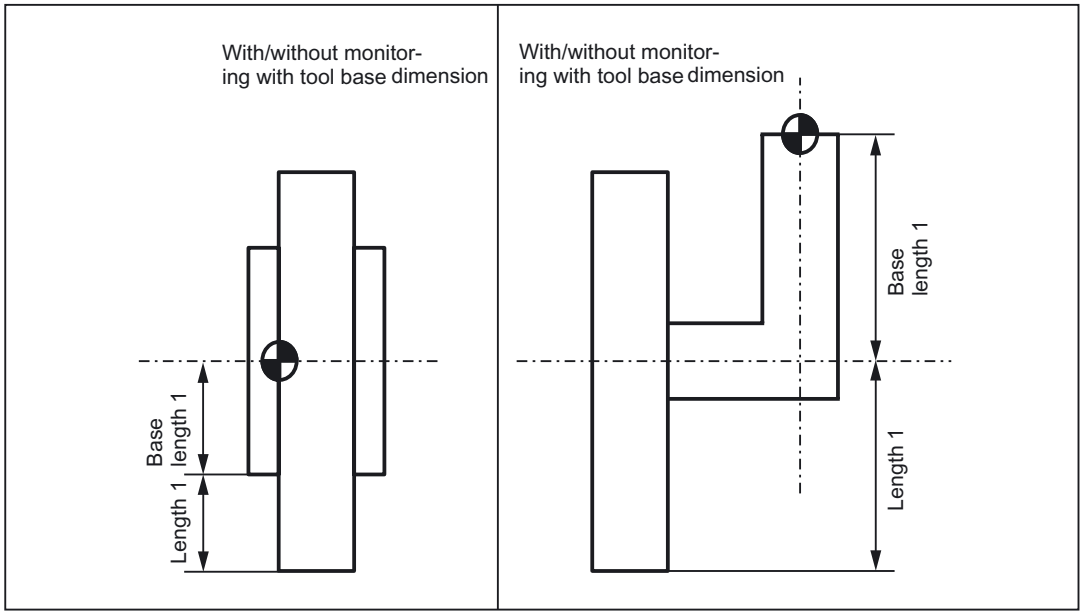
18.2.2 Cutting-edge-specific offset data

Tool parameter

The tool parameters for grinding tools have the same meaning as those for turning and milling tools.

Tool parameter	Description	Comment	
1	Tool type		
2	Cutting edge position	For turning and grinding tools only	
Geometry tool length compensation			
3	Length 1		
4	Length 2		
5	Length 3		
Geometry tool radius compensation			
6	Radius 1		
7			Reserved[1)]
8			Reserved[1)]
9			Reserved[1)]
10			Reserved[1)]
11			Reserved[1)]
Wear tool length compensation			
12	Length 1		
13	Length 2		
14	Length 3		
Wear tool radius compensation			
15	Radius 1		
16			Reserved[1)]
17			Reserved[1)]
18			Reserved[1)]
19			Reserved[1)]
20			Reserved[1)]
Base dimension/adapter dimension tool length compensation			
21	Basic length 1		
22	Basic length 2		
23	Basic length 3		
Technology			
24	Undercut angle	only for turning tools	
25			Reserved[1)]

1) "Reserved" means that this tool parameter on the 840D is not used (reserved for expansions).



Note

The cutting edge data for D1 and D2 of a selected grinding tool can be chained, i.e. if a parameter in D1 or D2 is modified, then the same parameter in D1 or D2 is automatically overwritten with the new value (see tool-specific data $TC_TPG2).

Definition of additional parameters $TC_DPC1...10

For user-related cutting data, additional parameters:
$TC_DPC1 to 10
can be defined independently of the tool type using general machine data:
MD18096 $MN_MM_NUM_CC_TOA_PARAM

⚠ CAUTION

Changes to the MD take effect after POWER ON and will lead to initialization of the memory (back data up beforehand if necessary!).

Automatic changeover between grinding wheel offset left and right does **not** take place during contour grinding. This changeover must be programmed.

Tool types for grinding tools

The structure of tool types for grinding tools is as follows:

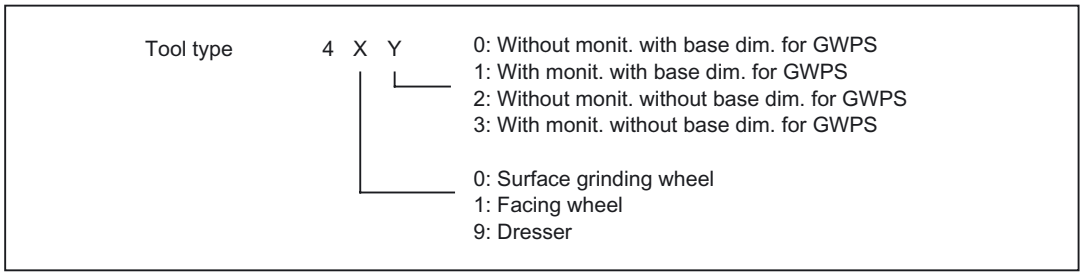


Figure 18-2 Structure of tool type for grinding tools

Note

MD20350 $MC_TOOL_GRIND_AUTO_TMON

Through this channel-specific machine data it can be determined, whether for grinding tools **with monitoring** (i.e. uneven tool types) the monitoring is already active or not when this tool is selected.

Examples:

This structure can be used to create the following tool types:

Type	Description
400	Surface grinding wheel
401	Surface grinding wheel with monitoring with base dimension for GWPS
403	Surface grinding wheel with monitoring without base dimension for GWPS
410	Facing wheel
411	Facing wheel with monitoring with base dimension for GWPS
413	Facing wheel with monitoring without base dimension for GWPS
490	Dresser

18.2.3 Tool-specific grinding data

Tool-specific grinding data

Tool-specific grinding data are available once for every T number (type 400- 499). They are automatically set up with every new grinding tool (type 400 - 499).

Note

Tool-specific grinding data have the same characteristics as a tool edge.

This is to be taken into account when the number of cuts is specified:

MD18100 $MN_MM_NUM_CUTTING_EDGES_IN_TOA

When all the cutting edges of a tool are deleted, the existing tool-specific grinding data are deleted at the same time.

Parameters

The parameters are assigned as follows:

Parameter	Meaning	Data type
$TC_TPG1	Spindle number	Integer
$TC_TPG2	Chaining rule	Integer
$TC_TPG3	Minimum wheel radius	Real
$TC_TPG4	Minimum wheel width	Real
$TC_TPG5	Current wheel width	Real
$TC_TPG6	Maximum speed	Real
$TC_TPG7	Maximum peripheral speed	Real
$TC_TPG8	Angle of inclined wheel	Real
$TC_TPG9	Parameter number for radius calculation	Integer
Additional parameters (user-specific cutting edge data)		
$TC_TPC1 to $TC_TPC10		Real

Definition of additional parameters $TC_DPC1...10

For the user-specific cutting data the additional parameters $TC_DPC1 to $TC_DPC10 can be implemented independent of the WZ-type. This is done via the general machine data:

MD18096 $MN_MM_NUM_CC_TDA_PARAM

⚠ CAUTION

Changes to the MD take effect after POWER ON and will lead to initialization of the memory (back data up beforehand if necessary!).

Spindle number $TC_TPG1

Number of programmed spindle (e.g. grinding wheel peripheral speed) and spindle to be monitored (e.g. wheel radius and width)

Chain rule $TC_TPG2

This parameter is set to define which tool parameters of tool edge 2 (D2) and tool edge 1 (D1) have to be chained to one another. When the setpoint of a chained parameter is modified, the value of the parameter with which it is chained is modified automatically.

Tool parameter	Meaning	Bit in $TC_TPG2	Hex	Dec
$TC_DP1	Tool type	0	0001	1
$TC_DP2	Length of cutting edge	1	0002	2
Geometry tool length compensation				
$TC_DP3	Length 1	2	0004	8
$TC_DP4	Length 2	3	0008	16
$TC_DP5	Length 3	4	0010	32
$TC_DP6	Radius	5	0020	64
$TC_DP7	Reserved	6	0040	128
$TC_DP8		7	0080	256
$TC_DP9		8	0100	512
$TC_DP10		9	0200	1024
$TC_DP11	Reserved	10	0400	2048
Wear tool length compensation				
$TC_DP12	Length 1	11	0800	4096
$TC_DP13	Length 2	12	1000	8192
$TC_DP14	Length 3	13	2000	16384
$TC_DP15	Radius	14	4000	32768
$TC_DP16	Reserved	15	8000	65536
$TC_DP17		16	10000	131072
$TC_DP18		17	20000	262144
$TC_DP19		18	40000	524288
$TC_DP20	Reserved	19	80000	1048576
Base dimension/adapter dimension tool length compensation				
$TC_DP21	Basic length 1	20	100000	2097152
$TC_DP22	Basic length 2	21	200000	4194304
$TC_DP23	Basic length 3	22	400000	8388608
Technology				
$TC_DP24	Reserved	23	800000	16777216
$TC_DP25	Reserved	24	1000000	33554432

Example of parameter chain:

Lengths 1, 2 and 3 of the geometry, the length wear and the tool base/adapter dimensions of lengths 1, 2 and 3 on a grinding tool (T1 in the example) must be automatically transferred.

Furthermore, the same tool type applies to tool edges 1 and 2.

Tool type	$TC_DP1	Bit 0
Length 1	$TC_DP3	Bit 2
Length 2	$TC_DP4	Bit 3
Length 3	$TC_DP5	Bit 4
Wear		
Length 1	$TC_DP12	Bit 11
Length 2	$TC_DP13	Bit 12
Length 3	$TC_DP14	Bit 13
Base/adapter dimension		
Length 1	$TC_DP21	Bit 20
Length 2	$TC_DP22	Bit 21
Length 3	$TC_DP23	Bit 22

Parameter $TC_TPG2 must therefore be assigned as follows:

Binary:	$TC_TPG2[1]= 'B111 0000 0011 1000 0001 1101' (Bit 22 ... Bit 0)
hexadecimal:	$TC_TPG2[1]= 'H70381D'
decimal:	$TC_TPG2[1]='D7354397'

Note

If the chaining specification is subsequently altered, the values of the two cutting edges are not automatically adjusted, but only after one parameter has been altered.

Minimum wheel radius and width $TC_TPG3 $TC_TPG4

The limit values for the grinding wheel radius and width must be entered in these parameters. These parameter values are used to monitor the grinding wheel geometry.

Note

It must be noted that the minimum grinding wheel radius must be specified in the Cartesian coordinate system for an inclined grinding wheel. A signal is output at the PLC interface if the grinding wheel width and radius drop below the minimum limits. The user can use these signals to define his error strategy.

Current width $TC_TPG5

The width of the grinding wheel measured, for example, after the dressing operation, is entered here.

Maximum speed and grinding wheel peripheral speed $TC_TPG6 $TC_TPG7

The upper limit values for maximum speed and peripheral speed of the grinding wheel must be entered in these parameters.

Requirement: A spindle has been declared.

Angle of inclined wheel $TC_TPG8

This parameter specifies the angle of inclination of an inclined wheel in the current plane. It is evaluated for GWPS.

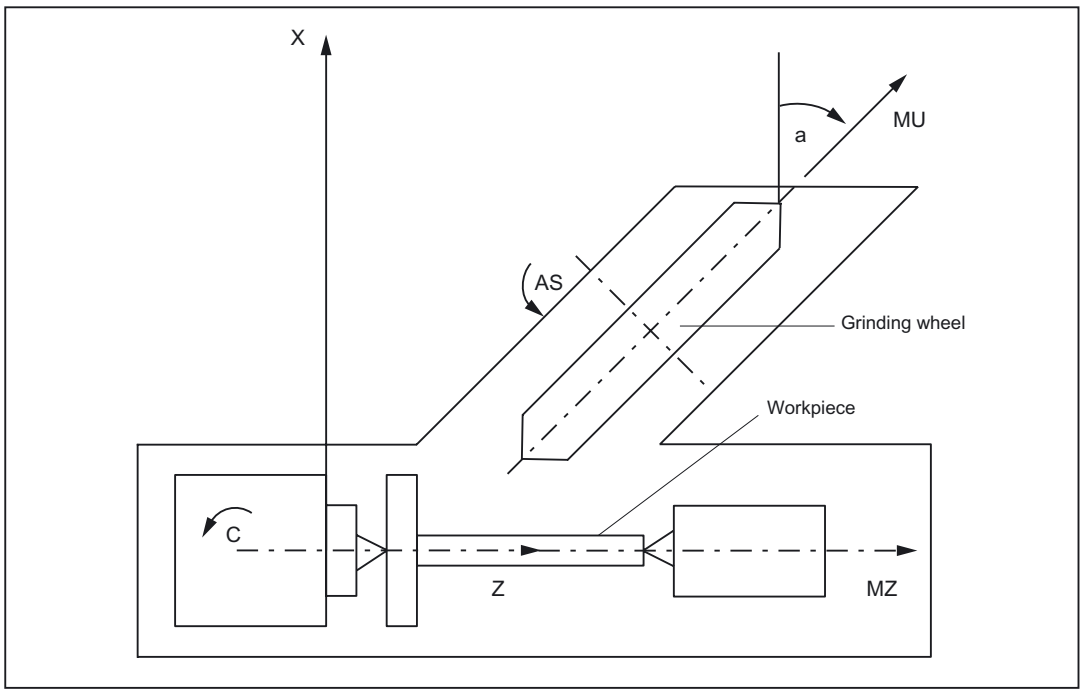


Figure 18-3 Machine with inclined infeed axis

Note

The tool lengths are not automatically compensated when the angle is altered.

The angle must be within the range -90° ≤ $TC_TPG8 < +90°.

On inclined axis machines the same angle must be specified for the inclined axis and the inclined wheel.

Parameter number for radius calculation $TC_TPG9

This parameter specifies which offset values are used for the GWPS calculation and tool monitoring of the minimum wheel radius ($TC_TPG3).

$TC_TPG9 = 3	Length 1 (geometry + wear + base, depending on tool type)
$TC_TPG9 = 4	Length 2 (geometry + wear + base, depending on tool type)
$TC_TPG9 = 5	Length 3 (geometry + wear + base, depending on tool type)
$TC_TPG9 = 6	Radius

Access from part program

Parameters can be read and written from the part program.

Example	Programming
Read the current width of tool 2 and store in R10	`R10 = $TC_TPG5 [2]`
Write value 2000 to the maximum speed of tool 3	`$TC_TPG6 [3] = 2000`

$P_ATPG[m] for current tool

This system variable allows the tool-specific grinding data for the **current** tool to be accessed.

m: Parameter number (data type: Real)

Example:

Parameter 3 ($TPG3[<T No.>])

$P_ATPG[3]=R10

Note

The monitoring data apply to both the left-hand and the right-hand cutting edge of the grinding wheel.

The tool-specific grinding data are activated when `GWPSON` (select constant grinding wheel surface speed) and `TMON` (select tool monitoring) are programmed. To activate a data which has been modified, it is necessary to program `GWPSON` or `TMON` again.

The length compensations always specify the distances between the tool carrier reference point and the tool tip in the Cartesian coordinates (must be noted for inclined grinding wheel).

18.2.4 Examples of grinding tools

Assignment of length offsets

Tool length compensations for the geometry axes or radius compensation in the plane are assigned on the basis of the current plane.

Planes

The following planes and axis assignments are possible (abscissa, ordinate, applicate for 1st, 2nd and 3rd geometry axes):

Command	Plane (abscissa/ordinate)	Axis perpendicular to plane (applicate)
G17	X/Y	Z
G18	Z/X	Y
G19	Y/Z	X

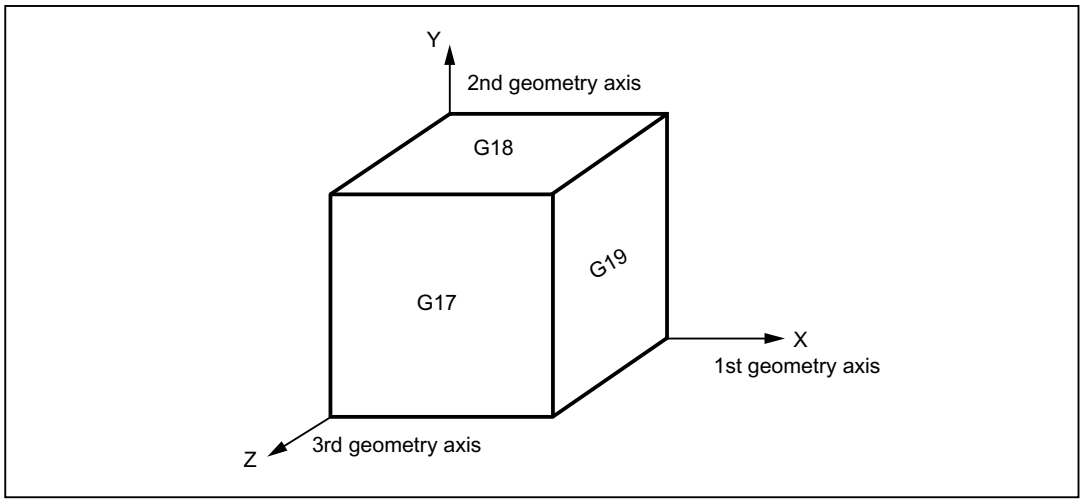


Figure 18-4 Planes and axis assignment

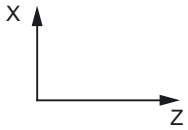
Surface grinding wheel

Entries in tool parameters

$TC_DP1	400
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius

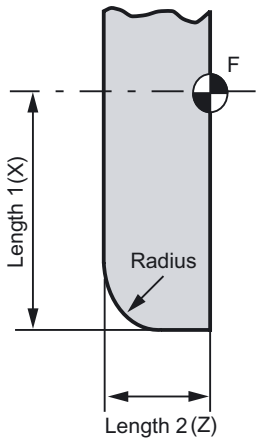
Unused parameters must be set to 0

e.g.
G18: Z/X plane



Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z



F: Tool holder reference point

Figure 18-5 Offset values required by a surface grinding wheel

Inclined wheel

Without tool base dimension for GWPS

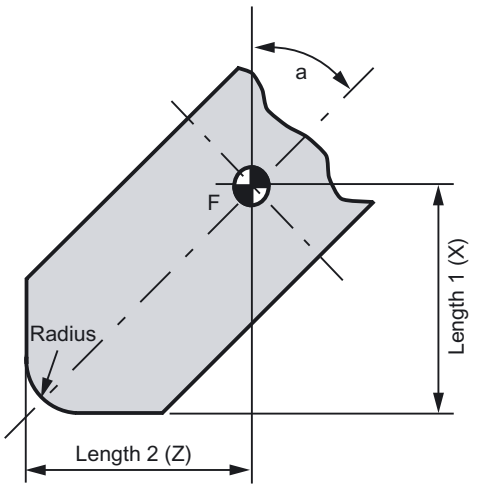
Entries in tool parameters

$TC_DP1	403
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z



F: Tool holder reference point

Figure 18-6 Offset values required for inclined wheel with implicit monitoring selection

Inclined wheel

With tool base dimension for GWPS

Entries in tool parameters	
$TC_DP1	401
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius
$TC_DP21	L1 base
$TC_DP22	L2 base

Wear values acc. to requirements

Unused parameters must be set to 0

Activation	
G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z

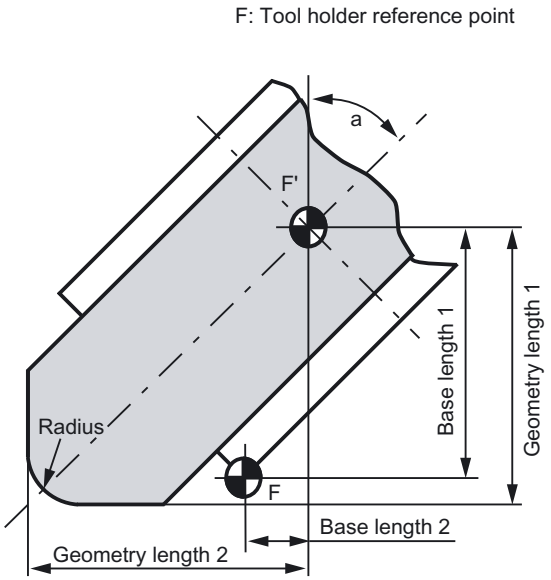


Figure 18-7 Required offset values shown by example of inclined grinding wheel with implicit monitoring selection and with base selection for GWPS calculation

Surface grinding wheel

Entries in tool parameters

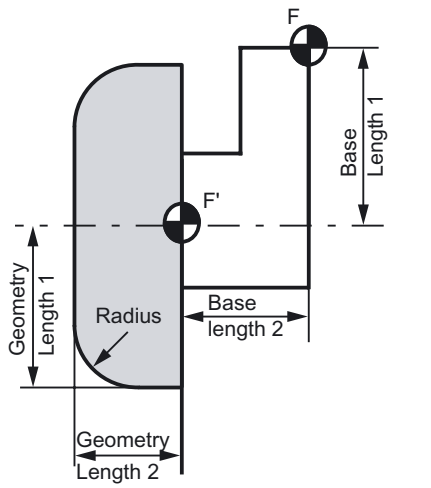
$TC_DP1	403
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius
$TC_DP21	L1 base
$TC_DP22	L2 base

Wear values acc. to requirements

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y Radius in Y/Z



F: Tool holder reference point

Figure 18-8 Required offset values of a surface grinding wheel without base dimension for GWPS

Facing wheel

Entries in tool parameters

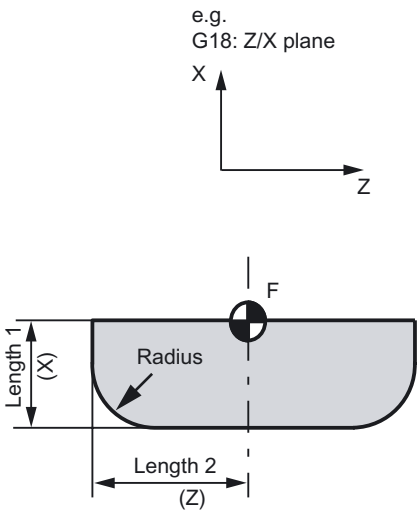
$TC_DP1	411
$TC_DP3	Length 1
$TC_DP4	Length 2
$TC_DP6	Radius

Wear values acc. to requirements

Unused parameters must be set to 0

Activation

G17	Length 1 in Y Length 2 in X Radius in X/Y
G18	Length 1 in X Length 2 in Z Radius in Z/X
G19	Length 1 in Z Length 2 in Y/Z



F: Tool holder reference point

Figure 18-9 Required offset values of a facing wheel with monitoring parameters

18.3 Online tool offset

18.3.1 General information

Application

A grinding operation involves both machining of a workpiece and dressing of the grinding wheel. These processes can take place in the same channel or in separate channels.

To allow the wheel to be dressed while it is machining a workpiece, the machine must offer a function whereby the reduction in the size of the grinding wheel caused by dressing is compensated on the workpiece. This type of compensation can be implemented by means of the "Online tool offset" (Continuous Dressing) function.

Dressing during machining process

To allow machining to continue while the grinding wheel is being dressed, the reduction in the size of the grinding wheel caused by dressing must be transferred to the current tool in the machining channel as a tool offset that is applied immediately.

This parallel dressing operation can be implemented by means of the "Continuous Dressing (parallel dressing), Online tool offset" function.

Note

The online tool offset may only be used for grinding tools.

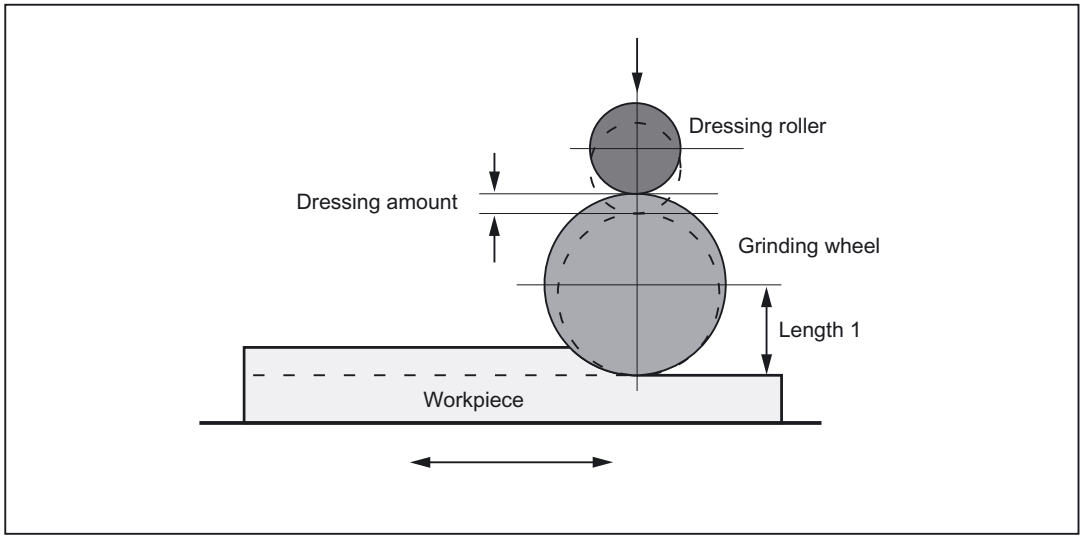


Figure 18-10 Dressing during machining using a dressing roller

General

An online tool offset can be activated for every grinding tool in any channel.

The online tool offset is generally applied as a length compensation. Like geometry and wear data, lengths are assigned to geometry axes on the basis of the current plane as a function of the tool type.

The grinding spindle monitoring function remains active when an online tool offset is selected.

Note

The offset always corrects the wear parameters of the selected length. If the length compensation is identical for several cutting edges, then a chaining specification must be used to ensure that the values for the 2nd cutting edge are automatically corrected as well.

If online offsets are active in the machining channel, then the wear values for the active tool in this channel may not be changed from the machining program or via operator inputs.

Modifications to the radius wear (P15) are not taken into account until the tool is reselected.

The online offset is also applied to the constant grinding wheel peripheral speed (GWPS), i.e. the spindle speed is corrected by the corresponding value.

Instructions

The following commands are provided for online tool offsets:

Command	Meaning
FCTDEF <polynomial no.>, <lower limit>, <upper limit>, <coefficient 0>, <coefficient 1>, <coefficient 2>, <coefficient 3>)	Parameterize function (up to 3rd degree polynomial) (Fine Tool Offset Definition)
PUTFTOCF (<polynomial no.>, <reference value>, <length1_2_3>, <channel no.>, <spindle no.>)	Write online tool offset continuously (Put Fine Tool Offset Compensation)
PUTFTOC (<value>, <length1_2_3>, <channel no.>, <spindle no.>)	Write online tool offset discretely (Put Fine Tool Offset Compensation)
FTOCON	Activation of online tool offset (Fine Tool Offset Compensation ON)
FTOCOF	Deactivation of online tool offset (Fine Tool Offset Compensation OFF)

Note

Changes to the correction values in the TOA memory do not take effect until T or D is programmed again.

References:
/PGA/ Programming Manual Job Planning

18.3.2 Write online tool offset: Continuous

FCTDEF

Certain dressing strategies (e.g. dressing roller) are characterized by the fact that the grinding wheel radius is continuously (linearly) reduced as the dressing roller is fed in. This strategy requires a linear function between infeed of the dressing roller and writing of the wear value of the respective length.

Function `FCTDEF` allows 3 independent functions to be defined according to the following syntax:

Function parameters

The function parameters are set in a separate block according to the following syntax:

FCTDEF(<polynomial no.>, <lower limit>, <upper limit>, <coefficient a0>, <coefficient a1>, <coefficient a2>, <coefficient a3>)

`FCTDEF`	Function definition
Polynomial no.:	Number of function (e.g. 1, 2 or 3)
Lower/upper limit:	Determines value range of the function (limit values in input resolutions)
Coefficients a_0, a_1, a_2:	Coefficients of polynomial

A 3rd degree polynomial is generally defined as follows:

$y = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$

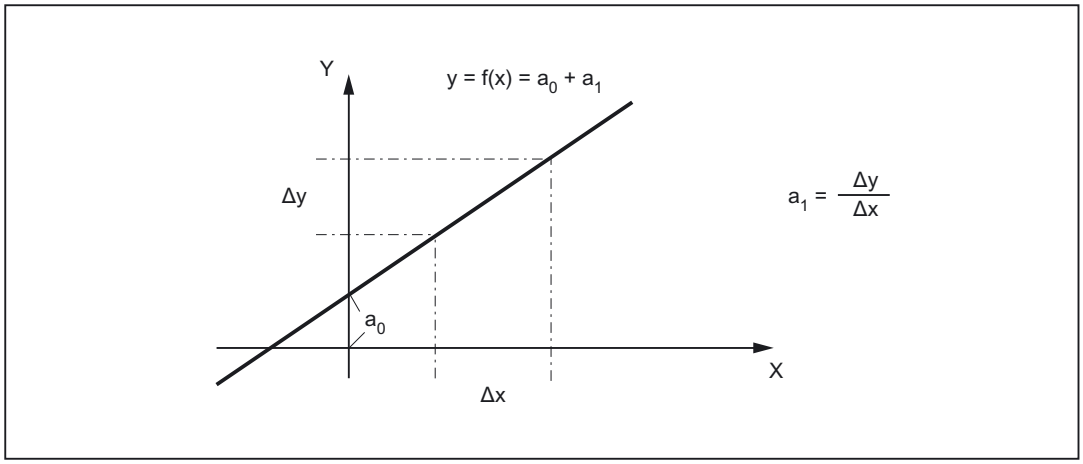


Figure 18-11 Straight line equation

Note

`FCTDEF` must be programmed in a separate NC block.

Example:

Existing conditions: Lead: $a_1 = +1$
$a_2 = 0$
$a_3 = 0$

At the time of definition, the function value y should be equal to 0 and should be derived from machine axis XA (e.g. dresser axis).

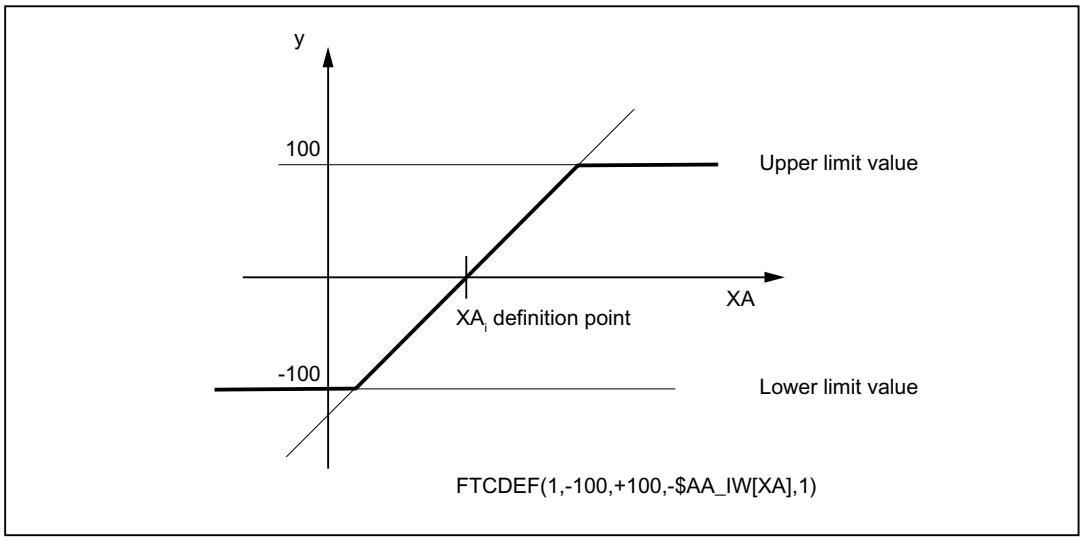


Figure 18-12 Straight line with gradient 1

Write online tool offset continuously

PUTFTOCF(<polynomial no.>, <reference value>, <length1_2_3>, <channel no.>, <spindle no.>)

PUFTOCF	
Polynomial no.:	Number of function (1, 2, 3)
Reference value:	Reference value of function
Length 1_2_3:	Wear parameter into which the tool offset value is added
Channel no.:	Channel in which the offset is to be effective
Spindle no.:	Spindle for which the online offset is to be effective

The online tool offset is activated before the dresser axis movement block.

Example:

```
FCTDEF(1, -100, 100, -$AA_IW[X], 1)          ;  Function definition
PUTFTOCF (1, $AA_IW[X], 1, 2, 1)             ;  Write online tool offset
                                                continuously
```

Length 1 of tool for spindle 1 in channel 2 is modified as a function of X axis movement.

Note

The online tool offset for a (geometric) grinding tool that is not active can be activated by specifying the appropriate spindle number.

If the channel number is omitted, the online offset is effective in the same channel.

If the spindle number is omitted, the online offset is applied to the current tool.

An online tool offset can also be called as a synchronized action.

References:
/FBSY/ Function Manual, Synchronized Actions

18.3.3 Activate/deactivate online tool offset

Activation/deactivation of online tool offset

The following commands activate and deactivate the online tool offset in the machining channel (grinding, destination channel):

`FTOCON` Activation of online tool offset

The machining channel can process online tool offsets (`PUTFTOC`) only if the offset is active (`FTOCON`). Alarm 20204 "PUTFTOC command not allowed" is otherwise output.

`FTOCOF` Deactivation of online tool offset

`FTOCOF` deactivates the online tool offset. The written values remain stored in the appropriate length compensation data.

Online offsets are traversed in the basic coordinate system, i.e. even when the workpiece coordinate system has been rotated, the length compensations always act in parallel to the coordinates of the unrotated system.
The offset is applied regardless of whether or not the axis to be compensated is traversed in the current block.

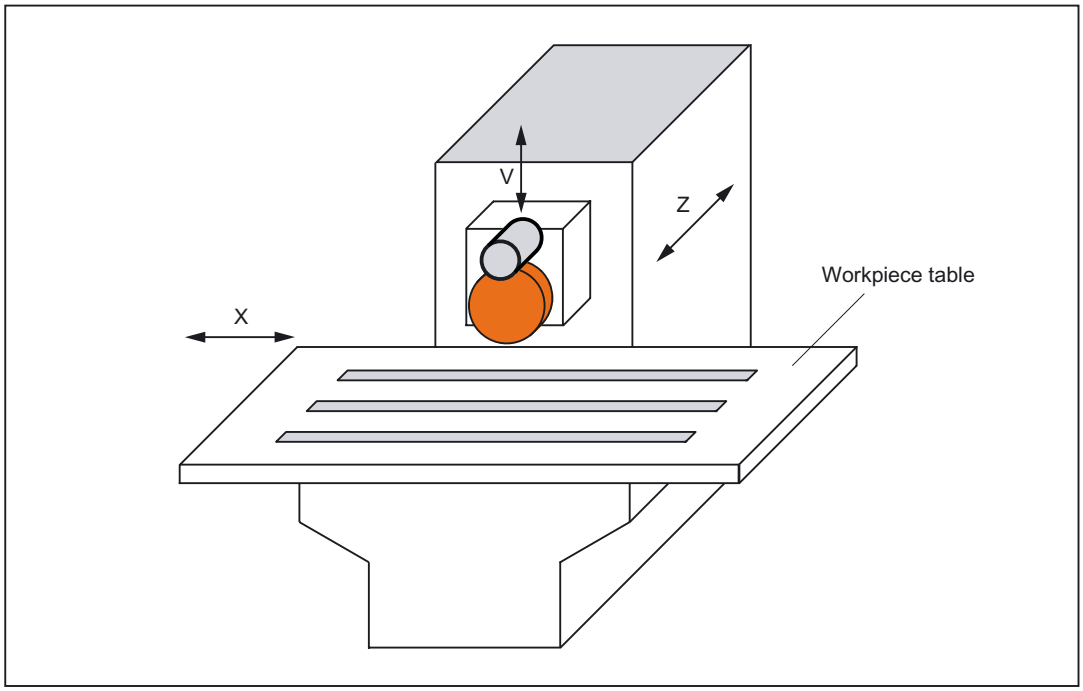
Note

Command `FTOCON` must be written to the channel in which the offset is to be applied (machining channel for grinding operation).

`FTOCOF` always corresponds to the reset position. `PUTFTOC` commands are effective only when the part program and `FTOCON` command are active.

18.3.4 Example of writing online tool offset continuously

Surface grinding machine



Y: Infeed axis for grinding wheel
V: Infeed axis for dressing roller
X: Reciprocating axis, left - right

Plane for the tool offset: `G19` (Y/Z plane)

Length 1 acts in Z, length 2 in Y, tool type = 401

Machining:	Channel 1 with axes Y, X
Dressing:	Channel 2 with axis V

Task

After the grinding operation has started at Y100, the grinding wheel must be dressed by 0.05 (in V direction). The dressing amount must be compensated continuously by means of an online offset.

Main machining program in channel 1

```
...
G1 G19 F10 G90                                  ;  Basic position
T1 D1                                           ;  Select current tool
S100 M3 Y100                                    ;  Spindle ON, traverse to
                                                   starting position
FTOCON                                          ;  Activate online offset
INIT (2, "/_N_MPF_DIR/_N_ABRICHT_MPF", "S")     ;  Select program in channel 2
START (2)                                       ;  Start program in channel 2

Y200                                            ;  Travel to target position
...
M30
```

Dressing program in channel 2_N_ABRICHT_MPF

```
...
FCTDEF (1, -1000, 1000, -$AA_IW[V], 1)          ;  Function definition
PUTFTOCF (1, $AA_IW[V], 2, 1)                   ;  Write online tool offset
                                                ;  continuously
U-0.05 G1 F0.01 G91                             ;  Infeed movement to dress wheel
...
M30
```

Note

Axis V operates (dresses) in parallel to Y, i.e. length 2 acts in Y and must therefore be compensated.

18.3.5 Write online tool offset discretely

PUTFTOC

This command writes an offset value by means of a program command.

PUTFTOC(<value>, <length1_2_3>, <channel no.>, <spindle no.>)

Put Fine Tool Offset Compensation

The wear of the specified length (1, 2 or 3) is modified online by the programmed value.

Note

The online tool offset for a (geometric) grinding tool that is not active can be activated by specifying the appropriate spindle number.

If the channel number is omitted, the online offset is effective in the same channel.

If the spindle number is omitted, the online offset is applied to the current tool.

18.3.6 Information about online offsets

Response in the case of tool change

- In cases where FTOCON has been active since the last tool or cutting edge change, preprocessing stop with resynchronization is initiated in the control when a tool is changed.
- Cutting edge changes can be implemented without preprocessing stop.

Note

Tool changes can be executed in conjunction with the online tool offset through the selection of T numbers.

Tool changes with M6 cannot be executed in conjunction with the online tool offset function.

Machining plane and transformation

- FTOCON can be used only in conjunction with the "Inclined axis" transformation.
- It is not possible to change transformations or planes (e.g. G17 to G18) when FTOCON is active, except in the FTOCOF state.

Resets and operating mode changes

- When online offset is active, NC-STOP and program end with M2/M30 are delayed until the amount of compensation has been traversed.
- The online tool offset is immediately deselected in response to NC-RESET.
- Online tool offsets can be activated in AUTOMATIC mode and when the program is active.

Supplementary conditions

- The online tool offset is superimposed on the programmed axis motion, allowing for the defined limit values (e.g. velocity).

  If a DRF offset and online offset are active simultaneously for an axis, the DRF offset is considered first.

- The pending offset is traversed at JOG velocity, allowing for the maximum acceleration.

  In case of `FTOCON` the following channel-specific machine data is taken into account:

  MD20610 $MC_ADD_MOVE_ACCEL_RESERVE

  An acceleration margin can thus be reserved for the movement which means that the overlaid movement can be executed immediately.

- The valid online offset is deleted on reference point approach with `G74`.
- The fine offset is not deselected for tool changes with `M6`.

18.4 Online tool radius compensation

General information

When the longitudinal axis of the tool and the contour are perpendicular to each other, the offset can be applied as a length compensation to one of the three geometry axes (online tool length compensation).

If this condition is not fulfilled, then the offset quantity can be entered as a real radius compensation value (online tool radius compensation).

Enabling of function

The online tool-radius offset is activated via the machine data:

MD20254 $MC_ONLINE_CUTCOM_ENABLE (enable online tool radius compensation).

Activation/deactivation

An online tool radius compensation is activated and deactivated by means of commands `FTOCON` and `FTOCOF`
(in the same way as an online tool length compensation).

Parameterization

The parameters of the online tool offset are set using commands `PUTFTOCF` and `PUTFTOC`.
Parameter "LENGTH 1_2_3" must be supplied as follows for an online tool radius compensation:

Parameter <length 1_2_3> = 4
Wear parameter to which correction value is added.

Supplementary conditions

- A tool radius compensation, and thus also an online tool radius compensation, can be activated only when the selected tool has a radius other than zero. This means that machining operations cannot be implemented solely with a tool radius compensation.
- The online offset values should be low in comparison to the original radius to prevent the permitted dynamic tolerance range from being exceeded when the offset is overlaid on the axis movement.
- When online tool radius compensation is applied to grinding and turning tools (types 400-599), the compensation value is applied as a function of the tool point direction, i.e. it acts as a radius compensation when tool radius compensation is active and as a length compensation when tool radius compensation is deactivated in the axes specified by the tool point direction.

  On all other tool types, the compensation value is applied only when tool radius compensation has been activated with `G41` or `G42`. The compensation value is canceled when tool radius compensation is deactivated with `G40`.

18.5 Grinding-specific tool monitoring

18.5.1 General information

Activation

The tool monitoring function is a combination of geometry and speed monitors and can be activated for any grinding tool (tool type: 400 to 499).

Selection

The monitoring function is selected:

- by programming (TMON) in the part program

  or

- automatically through selection of tool length compensation of a grinding tool with uneven tool type number.

Note

The automatic selection of the monitoring must be set via the channel-specific machine data:

MD20350 $MC_TOOL_GRIND_AUTO_TMON.

Monitoring active

The monitor for a grinding tool remains active until it is deselected again by means of program command TMOF.

Note

Monitoring of one tool is not deselected if the monitoring function is selected for another tool provided the two tools are referred to different spindles.

One tool and thus also **one** tool monitor can be active for every spindle at any point in time.

Activated monitors remain active after a RESET.

18.5.2 Geometry monitoring

Function

The following quantities can be monitored:

- The current grinding wheel radius

  and

- The current grinding wheel width

The current wheel radius is compared with the value stored in parameter $TC_TPG3.
The current radius is compared with the parameter number of the first edge (D1) of a grinding tool declared in parameter $TC_TPG9.

The current wheel width is generally calculated by the dressing cycle and can be entered in parameter $TC_TPG5 of a grinding tool. The value entered in this parameter is compared to the value stored in parameter $TC_TPG4 when the monitoring function is active.

When does monitoring take place?

The monitoring function for the grinding wheel radius remains active when an online tool offset is selected:

- When the monitoring function is activated
- when the current radius (online tool offset, wear parameter) or the current width ($TC_TPG5) is altered

Monitor reactions

If the current grinding wheel radius becomes smaller than the value stored in parameter $TC_TPG3 or the current grinding wheel width ($TC_TPG5) drops below the value defined in $TC_TPG4, the axis/spindle-specific bit DBX83.3 is set to "1" in DB31, ... at the PLC interface.
This bit is otherwise set to "0".

DB31, ... DBX83.3 = 1 ⇒ Geometry monitoring has responded

DB31, ... DBX83.3 = 0 ⇒ Geometry monitoring has not responded

Note

No error reaction is initiated internally in the control system.

18.5.3 Speed monitoring

Function

The speed monitor checks the grinding wheel peripheral speed (parameter $TC_TPG7) as well as the maximum spindle speed (parameter $TC_TPG6).

The unit of measurement is:

- Grinding wheel peripheral speed $m * s^{-1}$
- Spindle speed rev/min

Monitoring is cyclic. The value is always limited to the first limit value reached.

When does monitoring take place?

The speed setpoint is monitored against the speed limitation cyclically, allowing for the spindle override.

When is the speed limit value recalculated?

The speed limit value is recalculated:

- when the monitoring function is selected,
- when the online offset values (wear parameters) are altered.

Monitor reactions

The system reacts as follows when the speed monitor responds:

- The speed is restricted to the limit value

  and

- Interface signal:
  DB31, ... DBX83.6 (speed monitoring)
  is output.

DB31, ... DBX83.6 = 1 ⇒ Speed monitoring limit reached

DB31, ... DBX83.6 = 0 ⇒ Speed monitoring limit not reached

Note

No error reaction is initiated internally in the control system.

18.5.4 Selection/deselection of tool monitoring

Part program commands

The following part program commands are provided for selecting and deselecting the grinding-specific tool monitor of an active or inactive tool:

Command	Meaning
TMON Tool monitoring ON	Selection of tool monitoring for the active tool in the channel.
TMOF Tool monitoring OFF	Deselection of tool monitoring for the active tool in the channel.
TMON (T number) Tool monitoring ON (T No.)	Selection of tool monitoring for a non-active tool with T number.
TMOF (T number) Tool monitoring OFF (T No.)	Deselection of tool monitoring for a non-active tool with T number.
TMOF (0) Tool monitoring OFF (0)	Deselection of tool monitoring for all tools.

18.6 Constant grinding wheel peripheral speed (GWPS)

18.6.1 General information

What is GWPS?

A grinding wheel peripheral speed, as opposed to a spindle speed, is generally programmed for grinding wheels. This variable is determined by the technological process (e.g. grinding wheel characteristics, material pairing). The speed is then calculated from the programmed value and the current wheel radius.

Note

GWPS can be selected for grinding tools (types 400- 499).

Speed calculation

The formula for calculating the speed is as follows:

$$n\,[rpm] = \frac{GWP[m*s\text{-}1] * 60}{2\pi * R[m]}$$

Note

Grinding wheel peripheral speed can be programmed and selected for grinding tools (types 400- 499).

The wear is considered when calculating the radius (parameter $TC_TPG9).

This function also applies to inclined wheels/axes.

The associated wear and the base dimension as a function of the tool type are added to the parameter selected by $TC_TPG9.

The sum total is divided by "cos" ($TC_TPG8) when the value of parameter $TC_TPG8 (angle of inclined wheel) is positive, and is divided by "sin" ($TC_TPG8) when the value is negative.

When is the speed recalculated?

The speed is recalculated in response to the following events:

- GWPS programming
- Change in the online offset values (wear parameters).

18.6.2 Selection/deselection and programming of GWPS, system variable

Part program commands

The GWPS is selected and deselected with the following part program commands:

Command	Description
GWPSON Grinding wheel peripheral speed ON	Selection of GWPS for the active tool in the channel.
GWPSOF Grinding wheel peripheral speed OFF	Deselection of GWPS for the active tool in the channel.
GWPSON(T number) Grinding wheel peripheral speed ON (T no.)	Selection of GWPS for a non-active tool with T number.
GWPSOF(T number) Grinding wheel peripheral speed OFF (T no.)	Deselection of GWPS for a non-active tool with T number.
S[spindle number] = value	Programming of constant grinding wheel peripheral speed. Unit of value setting depends on basic system (m/s or ft/s).

References:
/PG/ Programming Manual Fundamentals

Note

Parameter $TC_TPG1 assigns a spindle to the tool. Every following S value for this spindle is interpreted as a grinding wheel peripheral speed when GWPS is active (see above).

If GWPS is to be selected with a new tool for a spindle for which the GWPS function is already active, the active function must be deselected first with GWPSOF (otherwise an alarm is given out).

GWPS can be active simultaneously for several spindles, each with a different grinding tool, in the same channel.

Selection of GWPS with GWPSON does not automatically result in activation of tool length compensation or of the geometry and speed monitoring functions. When GWPS is deselected, the last speed to be calculated remains valid as the setpoint.

$P_GWPS[spindle number]

This system variable can be used to query from the sub-program whether the GWPS is active for a specific spindle.

TRUE:	GWPS programming of spindle active
FALSE:	GWPS programming of spindle not active

References:
/PG/ Programming Manual Fundamentals

18.6.3 GWPS in all operating modes

General information

This function allows the constant grinding wheel peripheral speed (GWPS) function to be selected for a spindle immediately after `POWER ON` and to ensure that it remains active after an operating mode changeover, `RESET` or part program end.

The function is activated using machine data:
MD35032 $MA_SPIND_FUNC_RESET_MODE
(parameterization of GWPS function).

GWPS after POWER ON

A grinding-specific tool is defined via the following machine data:

MD20110 $MC_RESET_MODE_MASK

MD20120 $MC_TOOL_RESET_VALUE

MD20130 $MC_CUTTING_EDGE_RESET_VALUE

Note

MD35032 $MA_SPIND_FUNC_RESET_MODE
If the above machine data is set and a grinding-specific tool (tool type 400 to 499, MD20110, MD20120, MD20130) is used with reference to a valid spindle (parameter $TC_TPG1), then GWPS is activated for that spindle.

GWPS is deselected for all other spindles in this channel.

GWPS after RESET/part program end

After a RESET/part program end, GWPS remains active for all spindles for which it was already selected.

Note

MD35032 $MA_SPIND_FUNC_RESET_MODE
If the machine data above is set and GWPS is active on RESET or part program end, then GWPS remains active for this spindle.

If machine data:
MD35032 $MA_SPIND_FUNC_RESET_MODE is
not set and GWPS is active on RESET or part program end, then GWPS is deactivated for that spindle.

GWPS is deselected for all other spindles in this channel.

Machine data:
MD35040 $MA_SPIND_ACTIVE_AFTER_RESET
can be set to determine whether the spindle continues to rotate at the current speed after RESET.

Programming

The spindle speed can be modified through the input of a grinding wheel peripheral speed.

The spindle speed can be modified through:

- programming in the part program/overstoring
- programming the grinding wheel peripheral speed through assignment to address "S" in MDA
- spindle speed control via PLC (FC18).

DB31, ... DBX84.0 (GWPS active)

Interface signal:
DB31, ... DBX84.0 (GWPS active)
can be used to determine whether or not GWPS is active.

18.6.4 Example of how to program GWPS

Data of tool T1 (peripheral grinding wheel)

```
$TC_DP1[1,1] = 403              ;Tool type
$TC_DP3[1,1] = 300              ;Length1
$TC_DP4[1,1] = 50               ;Length2
$TC_DP12[1,1] = 0               ;Wear length 1
$TC_DP13[1,1] =0                ;Wear length 2
```

```
$TC_DP21[1.1] =300              ;Base length 1
$TC_DP22[1.1] =400              ;Base length 2
$TC_TPG1[1] = 1                 ;Spindle number
$TC_TPG8[1] = 0                 ;Angle of inclined wheel
$TC_TPG9[1] = 3                 ;Parameter no. for radius calculation
```

Data of tool T5 (inclined grinding wheel)

```
$TC_DP1[5,1] = 401              ;Tool type
$TC_DP3[5,1] = 120              ;Length1
$TC_DP4[5,1] = 30               ;Length2
$TC_DP12[5,1] = 0               ;Wear length 1
$TC_DP13[5,1] =0                ;Wear length 2
$TC_DP21[5,1] =100              ;Base length 1
$TC_DP22[5,1] =150              ;Base length 2
$TC_TPG1[5] = 2                 ;Spindle number
$TC_TPG8[5] = 45                ;Angle of inclined wheel
$TC_TPG9[5] = 3                 ;Parameter no. for radius calculation
```

Programming

```
N20     T1 D1                       ;  Select T1 and D1
N25     S1=1000 M1=3                ;  1000 rpm for spindle 1
N30     S2=1500 M2=3                ;  1500 rpm for spindle 2
...
N40     GWPSON                      ;  ;Selection of GWPS for active tool T1
N45     S[$P_AGT[1]] = 60           ;  Set GWPS to 60 m/s for active tool
                                    ;  n=1909.85 rpm
...
N50     GWPSON(5)                   ;  GWPS selection for tool 5 (2nd spindle)
N55     S[$TC_TPG1[5] ] = 40        ;  Set GWPS to 40 m/s for spindle 2
                                    ;  n=1909.85 rpm
...
N60     GWPSOF                      ;  Deactivate GWPS for active tool
N65     GWPSOF(5)                   ;  Switch off GWPS for tool 5 (spindle 2)
...
```

Supplementary references

/FB2/ Function Manual, Extended Functions; Oscillate (P5)

/FB1/ Function Manual Basic Functions; Feedrates (V1)

/FBSY/ Function Manual, Synchronized Actions

18.7 Supplementary conditions

18.7.1 Tool changes with online tool offset

Tool change

Tool changes with M6 cannot be executed in conjunction with the online tool offset function.

18.8 Data lists

18.8.1 Machine data

18.8.1.1 General machine data

Number	Identifier: $MN_	Description
18094	MM_NUM_CC_TDA_PARAM	Number of TDA
18096	MM_NUM_CC_TOA_PARAM	Number of TOA
18100	MM_NUM_CUTTING_EDGES_IN_TOA	Tool offsets per TOA

18.8.1.2 Channel-specific machine data

Number	Identifier: $MC_	Description
20254	ONLINE_CUTCOM_ENABLE	Enable online tool radius compensation
20350	TOOL_GRIND_AUTO_TMON	Automatic tool monitoring
20610	ADD_MOVE_ACCEL_RESERVE	Acceleration reserve for overlaid movements

18.8.1.3 Axis/spindle-specific machine data

Number	Identifier: $MA_	Description
32020	JOG_VELO	JOG axis velocity
35032	SPIND_FUNC_RESET_MODE	Parameterization of GWPS function

18.8.2 Signals

18.8.2.1 Signals from axis/spindle

Signal name	SINUMERIK 840D sl	SINUMERIK 828D
Geometry monitoring	DB31, ...DBX83.3	DB390x.DBX2001.3
Speed monitoring	DB31, ...DBX83.6	DB390x.DBX2001.6
GWPS active	DB31, ...DBX84.1	DB390x.DBX2002.1

Z2: NC/PLC interface signals

19

19.1 Digital and analog NCK I/Os

19.1.1 Signals to NC (DB10)

Overview of signals from PLC to NC

DB10	Signals to NC interface PLC → NC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Disable digital NCK inputs							
	Digital inputs without hardware *)				(on-board inputs) **)			
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
1	Setting on PLC of digital NCK inputs							
	Digital inputs without hardware *)				(on-board inputs) **)			
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
4	Disable digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
5	Overwrite mask for digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
6	Setting value from PLC for the digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
7	Setting mask for digital NCK outputs							
	Digital outputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
Notes: *) Bits 4 to 7 of the digital NCK outputs can be processed by the PLC even though there are no equivalent hardware I/Os. These bits can therefore also be used for data exchange between the NCK and PLC. **) With the 840D, the NCK digital inputs and outputs 1 to 4 are provided as onboard hardware inputs and outputs. These can be processed by the PLC according to *).								
DB10	Signals to NC interface PLC ! NC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
122	Disable digital NCK inputs							
	Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9

123	Setting on PLC of digital NCK inputs							
	Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9
124	Disable digital NCK inputs							
	Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17
125	Setting on PLC of digital NCK inputs							
	Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17
126	Disable digital NCK inputs							
	Input 32	Input 31	Input 30	Input 29	Input 28	Input 27	Input 26	Input 25
127	Setting on PLC of digital NCK inputs							
	Input 32	Input 31	Input 30	Input 29	Input 28	Input 27	Input 26	Input 25
128	Disable digital NCK inputs							
	Input 40	Input 39	Input 38	Input 37	Input 36	Input 35	Input 34	Input 33
129	Setting on PLC of digital NCK inputs							
	Input 40	Input 39	Input 38	Input 37	Input 36	Input 35	Input 34	Input 33
130	Disable digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
131	Overwrite mask for digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
132	Setting value from PLC for the digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
133	Setting mask for digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
134	Disable digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
135	Overwrite mask for digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
136	Setting value from PLC for the digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
137	Setting mask for digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
138	Disable digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
139	Overwrite mask for digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
140	Setting value from PLC for the digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25

141	Setting mask for digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
142	Disable digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
143	Overwrite mask for digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
144	Setting value from PLC for the digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
145	Setting mask for digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33
146	Disable analog NCK inputs							
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
147	Setting mask for analog NCK inputs							
	Input 8	Input 7	Input 6	Input 5	Input 4	Input 3	Input 2	Input 1
148, 149	Setting value from PLC for analog input 1 of the NCK							
150, 151	Setting value from PLC for analog input 2 of the NCK							
152, 153	Setting value from PLC for analog input 3 of the NCK							
154, 155	Setting value from PLC for analog input 4 of the NCK							
156, 157	Setting value from PLC for analog input 5 of the NCK							
158, 159	Setting value from PLC for analog input 6 of the NCK							
160, 161	Setting value from PLC for analog input 7 of the NCK							
162, 163	Setting value from PLC for analog input 8 of the NCK							
166	Overwrite mask for analog NCK outputs							
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
167	Setting mask for analog NCK outputs							
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
168	Disable analog NCK outputs							
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
170, 171	Setting value from PLC for analog output 1 of NCK							
172, 173	Setting value from PLC for analog output 2 of NCK							
174, 175	Setting value from PLC for analog output 3 of NCK							
176, 177	Setting value from PLC for analog output 4 of NCK							
178, 179	Setting value from PLC for analog output 5 of NCK							
180, 181	Setting value from PLC for analog output 6 of NCK							
182, 183	Setting value from PLC for analog output 7 of NCK							
184, 185	Setting value from PLC for analog output 8 of NCK							

Description of signals from PLC to NC

DB10 DBB0, 122, 124, 126, 128	**Disable digital NCK inputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The digital input of the NCK is disabled by the PLC. It is thus set to "0" in a defined way in the control.
Signal state 0 or edge change 1 → 0	The digital input of the NCK is enabled. The signal state applied at the input can now be read directly in the NC parts program.
Corresponding to	DB10 DBB1 (Setting by PLC of digital NCK inputs) DB10 DBB60 (actual value for digital NCK inputs) MD10350 $MN_FASTIO_DIG_NUM_INPUTS

DB10 DBB1, 123, 125, 127, 129	**Setting by PLC of digital NCK inputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The digital NCK input is set to a defined "1" state by the PLC. This means the signal state at the hardware input and disabling of the input (IS "Disable the digital NCK inputs") have no effect.
Signal state 0 or edge change 1 → 0	The signal state at the NCK input is enabled for read access by the NC parts program. However, the state can be accessed only if the NCK input is not disabled by the PLC (IS "Disable digital NCK inputs" = 0).
Corresponding to ...	DB10 DBB0 (Disable digital NCK inputs) DB10 DBB60 (actual value for digital NCK inputs) MD10350 $MN_FASTIO_DIG_NUM_INPUTS

DB10 DBB4, 130, 134, 138, 142	**Disable digital NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The digital NCK output is disabled. "0V" is output in a defined way at the hardware output.
Signal state 0 or edge change 1 → 0	The digital output of the NCK is enabled. As a result, the value set by the NC parts program or the PLC is output at the hardware output.
Corresponding to ...	DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB5 (Setting mask for digital NCK outputs) DB10 DBB6 (Setting by PLC of digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB5, 131, 135, 139, 143	Overwrite screen form for digital NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	On signal transition 0 → 1 the previous NCK value is overwritten by the setting value (IS "Setting value from PLC for digital NCK outputs"). The previous NCK value, which, for example, was directly set by the parts program, is lost. The signal status defined by the setting value forms the new NCK value.
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 →1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value (DB10, DBB6) is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms via the PLC user program must be avoided.
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Setting mask for digital NCK outputs) DB10 DBB6 (Setting value by PLC of digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB6, 132, 136, 140, 144	Setting by PLC of digital NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The signal status for the digital hardware output can be changed by the PLC with the setting value. There are two possibilities: • With the "Overwrite screen form": With signal transition 0 → 1 in the 'overwrite screen form' the PLC overwrites the previous 'NCK value' with the 'setting value'. This is the new 'NCK value'. • With the 'setting screen form': On signal state 1 in the "setting screen form", the "PLC value" is activated. The value used is the "setting value". With "setting value" "1", signal level 1 is put out at the hardware output. With "0", the output level is 0. The associated voltage values can be found in: **References:** /PHD/SINUMERIK 840D Configuration Manual (HW)
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 →1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value (DB10, DBB6) is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms via the PLC user program must be avoided.
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB5 (Setting mask for digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB7, 133, 137, 141, 145	Setting screen form for digital NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Instead of the NCK value, the PLC value is output at the digital hardware output. The PLC value must first be deposited in IS "Setting value from PLC for digital NCK outputs". The current NCK value is not lost.
Signal state 0 or edge change 1 → 0	The NCK value is output at the digital hardware output.
Special cases, errors, ...	**Note:** The PLC interface for the setting value (DB10, DBB6) is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms via the PLC user program must be avoided.
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB6 (Setting value by PLC of digital NCK outputs) MD10360 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB146	Disable analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The analog input of the NCK is disabled by the PLC. It is thus set to "0" in a defined way in the control.
Signal state 0 or edge change 1 → 0	The analog input of the NCK is enabled. This means that the analog value at the input can be read directly in the NC parts program if the setting screen form is set to 0 signal by the PLC for this NCK input.
Corresponding to ...	DB10 DBB147 (Setting screen form of analog NCK inputs) DB10 DBB148 (Setting by PLC of analog NCK inputs) DB10 DBB199 ... (Actual value of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB147	Setting screen form of analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The setting value from the PLC acts as the enabled analog value (IS "Setting value from PLC for analog NCK inputs").
Signal state 0 or edge change 1 → 0	The analog value at the NCK input is enabled for read access by the NC parts program However, the state can be accessed only if the NCK input is not disabled by the PLC (IS "Disable analog NCK inputs" = 0).
Corresponding to	DB10 DBB146 (Disable analog NCK inputs) DB10 DBB148 to 163 (Setting by PLC of analog NCK inputs) DB10 DBB199-209 (Actual value of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB148 - 163	Setting value from PLC for analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	With this setting value a defined analog value can be set by the PLC. With IS "Setting screen form of analog NCK inputs", the PLC selects whether the analog value at the hardware input or the setting value from the PLC is to be used as the enabled analog value. The setting value from the PLC becomes active as soon as IS "Setting screen form" is set to "1". The setting value from the PLC is specified as a fixed point number (16 bit value including sign) in 2's complement.
Corresponding to	DB10 DBB146 (Disable analog NCK inputs) DB10 DBB147 (Setting screen form of analog NCK inputs) DB10 DBB199-209 (Actual value of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB166	Overwrite screen form of analog NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	On signal transition 0 → 1 the previous NCK value is overwritten by the setting value (IS "Setting value from PLC for analog NCK outputs"). The previous NCK value which, for example, was directly set by the part program, is lost. The analog value specified by the PLC setting value forms the new NCK value.
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 →1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program.
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

DB10 DBB167	Setting screen form of analog NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Instead of the NCK value, the PLC value is output at the analog hardware output. The PLC value must first be stored in IS "Setting value from PLC for the analog NCK outputs". The current NCK value is not lost.
Signal state 0 or edge change 1 → 0	The NCK value is output at the analog hardware output.

DB10 DBB167	Setting screen form of analog NCK outputs
Special cases, errors, ...	**Note:** The PLC interface for the setting value is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program.
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

DB10 DBB168	Disable analog NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The analog output of the NCK is disabled. "0V" is output in a defined way at the hardware output.
Signal state 0 or edge change 1 → 0	The analog output of the NCK is enabled. As a result, the value set by the NC parts program or the PLC is output at the hardware output.
Corresponding to	DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

DB10 DBB170 - 185	Setting value from PLC for analog NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	With this setting value, the value for the analog hardware output can be changed by the PLC. There are two possibilities: • With the "Overwrite screen form": With signal transition 0 → 1 in the 'overwrite screen form' the PLC overwrites the previous 'NCK value' with the 'setting value'. This is the new "NCK value". • With the "setting screen form": On signal state 1 in the "setting creen form", the "PLC value" is activated. The value used is the 'setting value'. The setting value from the PLC is specified as a fixed point number (16 bit value including sign) in 2's complement.
Signal state 0 or edge change 1 → 0	As the interface signal is only evaluated by the NCK on signal transition 0 →1 it must be reset to "0" again by the PLC user program in the next PLC cycle.
Special cases, errors, ...	**Note:** The PLC interface for the setting value is used both by the overwrite screen form (for signal transition 0 → 1) and the setting screen form (for signal state 1). Simultaneous activation of the two screen forms must be avoided via the PLC user program.
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

19.1.2 Signals from NC (DB10)

Overview of signals from NC to PLC

DB10	Signals to NC interface NC → PLC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
60	Actual value for digital NCK inputs							
					(on-board inputs) **)			
					Input 4	Input 3	Input 2	Input 1
64	Setpoint for digital NCK outputs							
	Digital inputs without hardware *)				on-board outputs **)			
	Output 8	Output 7	Output 6	Output 5	Output 4	Output 3	Output 2	Output 1
186	Actual value for digital NCK inputs							
	Input 16	Input 15	Input 14	Input 13	Input 12	Input 11	Input 10	Input 9
187	Actual value for digital NCK inputs							
	Input 24	Input 23	Input 22	Input 21	Input 20	Input 19	Input 18	Input 17
188	Actual value for digital NCK inputs							
	Input 32	Input 31	Input 30	Input 29	Input 28	Input 27	Input 26	Input 25
189	Actual value for digital NCK inputs							
	Input 40	Input 39	Input 38	Input 37	Input 36	Input 35	Input 34	Input 33
190	Setpoint for digital NCK outputs							
	Output 16	Output 15	Output 14	Output 13	Output 12	Output 11	Output 10	Output 9
191	Setpoint for digital NCK outputs							
	Output 24	Output 23	Output 22	Output 21	Output 20	Output 19	Output 18	Output 17
192	Setpoint for digital NCK outputs							
	Output 32	Output 31	Output 30	Output 29	Output 28	Output 27	Output 26	Output 25
193	Setpoint for digital NCK outputs							
	Output 40	Output 39	Output 38	Output 37	Output 36	Output 35	Output 34	Output 33

Notes:
*) Bits 4 to 7 of the digital inputs and NCK outputs can be processed by the PLC although no equivalent hardware I/Os exist. These bits can therefore also be used for data exchange between the NCK
and PLC. **) With the 840D, the NCK digital inputs and outputs 1 to 4 are provided as onboard hardware inputs and outputs. These can be processed by the PLC according to *).

DB10	Signals to NC interface NC ! PLC							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
194, 195	Actual value for analog input 1 of NCK							
196, 197	Actual value for analog input 2 of NCK							
198, 199	Actual value for analog input 3 of NCK							
200, 201	Actual value for analog input 4 of NCK							
202, 203	Actual value for analog input 5 of NCK							
204, 205	Actual value for analog input 6 of NCK							
206, 207	Actual value for analog input 7 of NCK							
208, 209	Actual value for analog input 8 of NCK							
210, 211	Setpoint for analog output 1 of NCK							
212, 213	Setpoint for analog output 2 of NCK							
214, 215	Setpoint for analog output 3 of NCK							
216, 217	Setpoint for analog output 4 of NCK							
218, 219	Setpoint for analog output 5 of NCK							
220, 221	Setpoint for analog output 6 of NCK							
222, 223	Setpoint for analog output 7 of NCK							
224, 225	Setpoint for analog output 8 of NCK							

Description of signals from NC to PLC

DB10 DBB60, 186 - 189	Actual value for digital NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Signal level "1" is active at the digital hardware input of the NCK.
Signal state 0 or edge change 1 → 0	Signal level "0" is active at the digital hardware input of the NCK.
Special cases, errors, ...	The influence of interface signal: DB10 DBB0 (Disable digital NCK inputs) is ignored for the actual value.
Corresponding to	DB10 DBB0 (Disable digital NCK inputs) MD10350 $MN_FASTIO_DIG_NUM_INPUTS

DB10 DBB64, 190 - 193	Setpoint for digital NCK outputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The NCK value for the digital output currently set (setpoint) is "1".
Signal state 0 or edge change 1 → 0	The NCK value for the digital output currently set (setpoint) is "0".
Signal irrelevant for ...	This "setpoint" is only output to the hardware output under the following conditions: • Output is not disabled (IS "Disable digital NCK outputs") • PLC has switched to the NCK value (IS "Setting screen form for digital NCK inputs") As soon as these conditions are fulfilled, the "setpoint" of the digital output corresponds to the "actual value".
Corresponding to	DB10 DBB4 (Disable digital NCK outputs) DB10 DBB5 (Overwrite screen form for digital NCK outputs) DB10 DBB6 (Setting value by PLC of digital NCK outputs) DB10 DBB5 (Setting mask for digital NCK outputs) MD10310 $MN_FASTIO_DIG_NUM_OUTPUTS

DB10 DBB194 - 209	Actual value for analog NCK inputs
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The analog value applied to the analog NCK input is signalled to the PLC. The actual value is set as a fixed point number (16 bit value including sign) in 2's complement by the NCK.
Signal state 0 or edge change 1 → 0	The effect of the PLC on the analog value (e.g. with IS "Disable analog NCK inputs") is ignored.
Corresponding to	DB10 DBB146 (Disable analog NCK inputs) DB10 DBB147 (Setting screen form of analog NCK inputs) DB10 DBB148-163 (Setting by PLC of analog NCK inputs) MD10300 $MN_FASTIO_ANA_NUM_INPUTS

DB10 DBB210 - 225	**Setpoint for analog NCK outputs**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The current set NCK value for the analog output (setpoint) is signalled to the PLC. The set value is set as a fixed point number (16 bit value including sign) in 2's complement by the NCK.
Signal state 0 or edge change 1 → 0	This 'setpoint' is only output to the hardware output under the following conditions: • Output is not disabled (IS "Disable analog NCK outputs") • The PLC has switched to the NCK value (IS "Setting screen form of analog NCK outputs")
Corresponding to	DB10 DBB168 (Disable analog NCK outputs) DB10 DBB166 (Overwrite screen form of analog NCK outputs) DB10 DBB170-185 (Setting by PLC of analog NCK outputs) DB10 DBB167 (Setting screen form of analog NCK outputs) MD10310 $MN_FASTIO_ANA_NUM_OUTPUTS

19.2 Several operator panels on several NCUs, distributed systems

19.2.1 Defined logical functions/defines

BUSTYP

Name	Value	Interface DB19	Meaning
MPI	1	DBW100,102,104, 120, 130 Bits 8 -15	Control unit to MPI, 187.5 kbaud
OPI	2	"	Control unit to MPI, 1.5 Mbaud

STATUS

Name	Value	Interface DB19	Meaning
OFFL_REQ_PLC	1	Online interfaces 1st: DBB124 2nd: DBB134	PLC to control unit: PLC wants to displace control unit by offline request.
OFFL_CONF_PLC	2	Online interfaces 1st: DBB124 2nd: DBB134	Control unit to PLC: Acknowledgement of OFFL_REQ_PLC The meaning of the signal is dependent on Z_INFO DBB125 or DBB135
OFFL_REQ_OP	3	Online interfaces 1st: DBB124 2nd: DBB134	Control unit to PLC: Control unit would like to go offline from this NCU and outputs an offline request
OFFL_CONF_OP	4	Online interfaces 1st: DBB124 2nd: DBB134	PLC to control unit: Acknowledgement of OFFL_REQ_OP The meaning of the signal is dependent on Z_INFO DBB125 or DBB135
ONL_PERM	5	Online request interface DBB108	PLC to control unit: PLC notifies control unit as to whether it can go online or not. The meaning of the signal is dependent on Z_INFO: DBB109
S_ACT	6	Online interfaces 1st: DBB124 2nd: DBB134	Control unit to PLC: Control unit goes online or changes operating focus. The meaning of the signal is dependent on Z_INFO DBB125 or DBB135
OFFL_REQ_FOC	7	Online interfaces 1st: DBB124 2nd: DBB134	Control unit to PLC: Control unit would like to take operating focus away from this NCU
OFFL_CONF_FOC	8	Online interfaces 1st: DBB124 2nd: DBB134	PLC to control unit: Acknowledgement of OFFL_REQ_FOC The meaning of the signal is dependent on Z_INFO DBB125 or DBB135

Name	Value	Interface DB19	Meaning
ONL_REQ_FOC	9	Online interfaces 1st: DBB124 2nd: DBB134	Control unit to PLC: Control unit would like to set operating focus to this NCU
ONL_PERM_FOC	10	Online interfaces 1st: DBB124 2nd: DBB134	PLC to control unit: Acknowledgement of ONL_REQ_FOC The meaning of the signal is dependent on Z_INFO DBB125 or DBB135

Z_INFO

Name	Value	Interface DB19	Meaning
DISC_FOC	9	DBB125 DBB135	Control unit switches operating focus to another NCU.
OK	10	DBB109 Bit 0-3 DBB125 DBB135	Positive acknowledgement
CONNECT	11	DBB125 DBB135	Control unit has gone online on this NCU.
MMC_LOCKED	13	DBB109 Bit 0-3 DBB125 DBB135	HMI has set switchover disable. There are processes running on this control unit that may not be interrupted by a switchover.
PLC_LOCKED	14	DBB109 Bit 0-3 DBB125 DBB135	The HMI switchover disable is set in the HMI-PLC interface. Control unit cannot go offline from this NCU or change operating focus.
PRIO_H	15	DBB109 Bit 0-3 DBB125 DBB135	Control units with a higher priority are operating on this NCU. Requesting control unit cannot go online to this NCU.

STATUS and Z_INFO can be combined as follows

Name: Status	Z_INFO	Meaning
OFFL_REQ_PLC	OK	PLC wants to displace online control unit by offline request.
OFFL_CONF_PLC	OK	Control unit positively acknowledges the offline request from PLC. Control unit will subsequently go offline.
OFFL_CONF_PLC	MMC_LOCKED	Control unit negatively acknowledges the offline request. Control unit will not go offline, as processes are running that must not be interrupted.
OFFL_REQ_OP	OK	Control unit would like to go offline from the online NCU and outputs an offline request.
OFFL_CONF_OP	OK	PLC positively acknowledges the offline request. Control unit will subsequently go offline from this NCU.

Name: Status	Z_INFO	Meaning
OFFL_CONF_OP	PLC_LOCKED	PLC negatively acknowledges the offline request from control unit. User has set the HMI switchover disable, control unit cannot go offline, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, 1st or 2nd HMI-PLC interface.
ONL_PERM	No. of HMI-PLC online interface, OK	PLC issues the online enabling command to the requesting control unit. Control unit can then go online to this NCU. Content of Z_INFO: Bit 0 ..3: OK Bit 4... 7: No. of HMI-PLC online interface with which the control unit should connect: First HMI-PLC online interface Second HMI-PLC online interface
ONL_PERM	MMC_LOCKED	The requesting control unit cannot go online. Two control units on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot suppress either of the two control units.
ONL_PERM	PLC_LOCKED	The requesting control unit cannot go online. User has set HMI switchover disable, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, first or second HMI online interface.
ONL_PERM	PRIO_H	The requesting control unit cannot go online. Two control units that are both higher priority than the requesting control unit are connected online to the NCU. The PLC cannot suppress either of the two control units.
S_ACT	CONNECT	The requesting MMC has gone online. The PLC now activates HMI sign-of-life monitoring.
S_ACT	DISC_FOCUS	Server HMI has disconnected the operating focus from this NCU.
OFFL_REQ_FOC	OK	Server HMI would like to disconnect the operating focus from this NCU and outputs an offline focus request.
OFFL_CONF_FOC	OK	PLC positively acknowledges the offline focus request. Server HMI can disconnect operating focus.
OFFL_CONF_FOC	PLC_LOCKED	PLC negatively acknowledges the online focus request. User has set HMI switchover disable, server HMI cannot disconnect operating focus, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, first or second HMI-PLC interface.
ONL_REQ_FOC	OK	Server HMI would like to set the operating focus on this NCU and outputs an online focus request.
ONL_PERM_FOC	OK	PLC positively acknowledges the online focus request. Server HMI then connects operating focus to this NCU.
ONL_PERM_FOC	PLC_LOCKED	PLC negatively acknowledges the online focus request. User has set HMI switchover disable, server HMI cannot set operating focus, MMCx_SHIFT_LOCK = TRUE, x=1 or 2, first or second HMI-PLC interface.

19.2.2 Interfaces in DB19 for M:N

The HMI-PLC interface in DB19 is divided into 3 areas

Online request interface

The online request sequence is executed on this interface if a control unit wants to go online.

HMI writes its client ID to ONL_REQUEST and waits for the return of the client ID in ONL_CONFIRM.

After the positive acknowledgement from the PLC, the control unit sends its parameters and waits for online permission (in PAR_STATUS, PAR_Z_INFO).

HMI parameter transfer:

Client identification -> PAR_CLIENT_IDENT

HMI type -> PAR_MMC_TYP

MCP address -> PAR_MSTT_ADR

With the positive online permission, the PLC also sends the number of the HMI-PLC online interface DBB109.4-7 to be used by the control unit.

The MMC then goes online and occupies the online interface assigned by the PLC.

Online interfaces

Two control units can be connected online to one NCU at the same time.

The online interface is available for each of the two online control units separately.

After a successful online request sequence, the control unit receives the number of its online interface from the PLC.

The HMI parameters are then transferred to the corresponding online interface by the PLC.

The control unit goes online and occupies its own online interface via which data are then exchanged between the HMI and PLC.

HMI data interfaces

User data from/to the HMI are defined on these:

- DBB 0-49 control unit 1 interface
- DBB 50-99 control unit 2 interface

These data and signals are always needed to operate control units.

M:N sign-of-life monitoring

This is an additional monitoring function which must not be confused with the HMI sign-of-life monitor. For further information, please refer to the relevant signals.

In certain operating states, control units with activated M:N switchover (parameterizable in NETNAMES.INI) must be capable of determining from a PLC data whether they need to wait or not before linking up with an NCU.

Example:

Control units with an activated control unit switchover function must be capable of starting up an NCU without issuing an online request first.

Control unit must go online for service-related reasons.

The operation is coordinated in the online request interface via data DBW110:
M_TO_N_ALIVE

The M:N sign of life is a ring counter which is incremented cyclically by the PLC or set to a value of 1 when it overflows.

Before a control unit issues an online request, it must check the sign of life to establish whether the M:N switchover is activated in the PLC.

Procedure:

HMI reads the sign of life at instants T0 and T0 + 1.

Case 1: negative acknowledgement for reading process, DB19 does not exist. Control unit goes online without request procedure.

Case 2: m_to_n_alive = 0, control unit switchover disabled. Control unit goes online without request procedure.

Case 3: m_to_n_alive (T0) = m_to_n_alive (T0+1), control unit switchover disabled. Control unit goes online without request procedure.

Case 4: m_to_n_alive (T0) <> m_to_n_alive (T0+1), control unit switchover enabled.

Case 1 ... case 3 apply only under special conditions and not in normal operation.

Online request interface

DB19 DBW100	**ONL_REQUEST**
Client_Ident	Control unit would like to go online and use the online request interface. HMI first writes its Client_Ident as a request. Bit 8 .. 15: Bus type: MPI 1 or BTSS 2 Bit 0 .. 7: HMI bus address
DB19 DBW102	**ONL_CONFIRM.**
Client_Ident	If the online request interface is not being used by another control unit, the PLC returns the Client identification as positive acknowledgement. Bit 8 .. 15: Bus type: MPI 1 or BTSS 2 Bit 0 .. 7: HMI bus address
DB19 DBW104	**PAR_CLIENT_IDENT** HMI parameter transfer to PLC
Client_Ident	Bit 8 .. 15: Bus type: MPI 1 or BTSS 2 Bit 0 .. 7: HMI bus address

DB19 DBB106	**PAR_MMC_TYP** HMI parameter transfer to PLC
HMI type from NETNAMES.INI	Type properties of the control unit configured in file NETNAMES.INI. Evaluated by the PLC when MMC is suppressed (server, main/secondary operator panel, ...), see description of file NETNAMES.INI
DB19 DBB107	**PAR_MSTT_ADR** HMI parameter transfer to PLC
MCP address from NETNAMES.INI	Address of MCP to be switched over or activated/deactivated with the control unit. Parameter from NETNAMES.INI
255	No MCP is assigned to control unit, no MCP will be activated/deactivated
DB19 DB108	**PAR_STATUS** PLC sends HMI pos./neg. online permission
ONL_PERM (5)	PLC notifies HMI as to whether control unit can go online or not. The meaning of the signal is dependent on PAR_Z_INFO:
DB19 DBB109	**PAR_Z_INFO** PLC sends HMI pos./neg. online permission
No. of HMI-PLC online interface, OK (10)	PLC issues the online enabling command to the requesting control unit. Control unit can then go online to this NCU. Bit 0 ..3: OK Bit 4 .. 7: No. of HMI-PLC online interface with which the control unit should connect: First HMI-PLC online interface Second HMI-PLC online interface
MMC_LOCKED (13)	The requesting control unit cannot go online. Two control units on which uninterruptible processes are in progress are connected online to this NCU. The PLC cannot suppress either of the two control units.
PLC_LOCKED (14)	The control unit switchover disable is set in the HMI-PLC interface.
PRIO_H (15)	The requesting control unit cannot go online. Two control units that are both higher priority than the requesting control unit are connected online to the NCU. The PLC cannot suppress either of the two control units.

Sign of life of M:N switchover

DB19 DBW110	**M_TO_N_ALIVE**
1 ... 65535	Ring counter that is cyclically incremented by the PLC. Indicator for the HMI that the M:N switchover is active and ready.

1st HMI-PLC online interface

DB19 DBW120	**MMC1_CLIENT_IDENT**
	refer to PAR_CLIENT_IDENT after issuing positive online permission, the PLC transfers the HMI parameters to the online interface PAR_CLIENT_IDENT -> MMC1_CLIENT_IDENT
DB19 DBB122	**MMC1_TYP**
	refer to PAR_MMC_TYP After issuing positive online permission, the PLC transfers the HMI parameters to the online interface PAR_MMC_TYP -> MMC1_TYP

DB19 DBB123	**MMC1_MSTT_ADR**
	refer to PAR_ MSTT_ADR After issuing positive online permission, the PLC transfers the HMI parameters to the online interface PAR_ MSTT_ADR -> MMC1_MSTT_ADR
DB19 DBB124	**MMC1_STATUS**
	Requests from online HMI to PLC or vice-versa. The meaning of the signal is dependent on MMC1_Z_INFO
OFFL_REQ_PLC (1)	PLC to HMI: PLC wants to displace control unit by offline request.
OFFL_CONF_PLC (2)	HMI to PLC: Acknowledgement of OFFL_REQ_PLC
OFFL_REQ_OP (3)	HMI to PLC: Control unit would like to go offline from this NCU and outputs an offline request
OFFL_CONF_OP (4)	PLC to HMI: Acknowledgement of OFFL_REQ_OP
S_ACT (6)	HMI to PLC: Control unit goes online or changes operating focus
OFFL_REQ_FOC (7)	HMI to PLC: Control unit would like to take operating focus away from this NCU
OFFL_CONF_FOC (8)	PLC to HMI: Acknowledgement of OFFL_REQ_FOC
ONL_REQ_FOC (9)	HMI to PLC: Control unit would like to set operating focus to this NCU
ONL_PERM_FOC (10)	PLC to HMI: Acknowledgement of ONL_REQ_FOC
DB19 DBB125	**MMC1_Z_INFO**
	Request from online HMI to PLC or vice-versa. The meaning of the signal is dependent on MMC1_STATUS
DISC_FOC (9)	Control unit switches operating focus to another NCU.
OK (10)	Positive acknowledgement
CONNECT (11)	Control unit has gone online on this NCU.
PLC_LOCKED (14)	The control unit switchover disable is set in the HMI-PLC interface. Control unit cannot go offline from this NCU or change operating focus.
PRIO_H (15)	Control units with a higher priority are operating on this NCU. Requesting control unit cannot go online to this NCU

Bit signals

DB19 **DBX 126.0** **Data Block**	**MMC1_SHIFT_LOCK** **Disable/enable control unit switchover**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Control unit switchover or change in operating focus is disabled. The current control unit-NCU constellation remains unchanged.
Signal state 0 or edge change 1 → 0	Control unit switchover or change in operating focus is enabled

DB19 DBX 126.1 Data Block	MMC1_MSTT_SHIFT_LOCK Disable/enable MCP switchover
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	MCP switchover is disabled. The current MCP-NCU constellation remains unchanged.
Signal state 0 or edge change 1 → 0	MCP switchover is enabled

DB19 DBX 126.2 Data Block	MMC1_ACTIVE_REQ Control unit 1 requests active operating mode
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	HMI to PLC: passive control unit 1 requests active operating mode
Signal state 0 or edge change 1 → 0	PLC to HMI: Request received

DB19 DBX 126.3 Data Block	MMC1_ACTIVE_PERM Active/passive operating mode
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	PLC to HMI: passive operator unit 1 can change to active operating mode
Signal state 0 or edge change 1 → 0	PLC to MMC: active operator panel must change to passive operating mode

DB19 DBX 126.4 Data Block	MMC1_ACTIVE_CHANGED Active/passive operating mode of MMC
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	HMI to PLC: Control unit has completed changeover from passive to active mode
Signal state 0 or edge change 1 → 0	HMI to PLC: Control unit has completed changeover from active to passive mode

DB19 DBX126.5 Data Block	MMC1_CHANGE_DENIED Operating mode changeover rejected
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	HMI to PLC or PLC to HMI depending on status of interface: Operating mode cannot be changed due to uninterruptible processes on active control unit.
Signal state 0 or edge change 1 → 0	HMI to PLC or PLC to HMI depending on status of interface: Acknowledgement on MMC1_CHANGE_DENIED (FALSE → TRUE)

2nd HMI-PLC online interface

The signals of the 2nd HMI-PLC online interface are analogous in meaning to the signals of the 1st HMI-PLC online interface (MMC2_ ... replaces MMC1_...)..

Sign-of-life monitoring HMI

After a control unit has gone online to an NCU, the HMI sign of life is set in the interface. (E_BTSSReady, E_MMCMPI_Ready, E_MMC2Ready)

The signals are automatically set by the HMI when the control unit goes online and stay set for as long as it remains online.

They are provided separately for each HMI-PLC interface and used by the PLC to monitor the HMI sign of life.

First HMI-PLC online interface

A distinction between an control unit link via the OPI (1.5 Mbaud) or the MPI (187.5 kbaud) is made on this interface.

The signal corresponding to the bus type is set while the control unit is online.

DB**10** DBX104.0	MCP1 ready
FALSE	MCP1 is not ready
TRUE	MCP1 is ready
DB**10** DBX104.1	MCP2 ready
FALSE	MCP2 is not ready
TRUE	MCP2 is ready
DB**10** DBX104.2	HHU ready
FALSE	HHU is not ready
TRUE	HHU is ready
DB**10** DBX108.3	E_MMCBTSSReady
FALSE	No control unit online to OPI
TRUE	Control unit online to OPI
DB**10** DBX108.2	E_MMCMPIReady
FALSE	No control unit online to MPI
TRUE	Control unit online to MPI

Second HMI-PLC online interface

This interface utilizes a group signal for both bus types. No distinction is made between OPI and MPI.

DB10 DBX108.1	E_MMC2Ready
FALSE	no control unit online to OPI or MPI
TRUE	Control unit online to OPI or MPI

The sign-of-life monitor is switched on by the PLC as soon as a control unit has gone online to its interface and switched off again when it goes offline.

Sign-of-life monitoring will be **enabled:**
as soon as control unit or HMI logs on online to its HMI-PLC interface with S_ACT/CONNECT.

Sign-of-life monitoring will be **disabled:**
as soon as control unit goes offline.

1. HMI wants to switch over and log off from PLC with

   OFFL_REQ_OP/OK

2. PLC acknowledges to HMI with OFFL_CONF_OP/OK
3. Control unit or HMI will be displaced by PLC with OFFL_REQ_PLC/OK

   HMI acknowledges to PLC with OFFL_CONF_PLC/OK

In both instances the PLC detects that a control unit is going offline and waits for the TRUE-FALSE edge of its HMI sign-of-life signal.

The PLC then ceases to monitor the sign-of-life signal.

19.2.3 Signals from NC (DB10)

DB10 DBX107.6	**NCU link active**
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	NCU link communication is active.
Signal state 0 or edge change 1 → 0	No NCU link communication is active.
Signal irrelevant for ...	Systems without NCU link modules.
References	/PHD/ NCU Configuration Manual

19.2.4 Signals from axis/spindle (DB31, ...)

DB31, ... DBX60.1	**NCU link axis active**
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	Axis is active as NCU link axis.
Signal state 0 or edge change 1 → 0	Axis is used as a local axis.
Signal irrelevant for ...	Systems without NCU link modules.
Additional references	/PHD/ NCU Configuration Manual

DB31, ... DBX61.1	**Axis container rotation active**
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	An axis container rotation is active for the axis.
Signal state 0 or edge change 1 → 0	An axis container rotation is not active for the axis.

DB31, ... DBX61.2	**Axis ready**
Edge evaluation:	Signal(s) updated:
Meaning	The signal is processed on the home NCU in the NCU link grouping. The home NCU is the NCU to which the axis is physically connected.
Signal state 1 or edge change 0 → 1	Axis is ready.
Signal state 0 or edge change 1 → 0	Axis is not ready. This status will be set when the channel, the operating modes group or the NCK have generated the alarm "not ready".

19.3 Operation via PG/PC (B4)

No signal descriptions required.

19.4 Manual and handwheel travel

19.4.1 Signals from NC (DB10)

<table>
<tr><td>DB10
DBB97, 98, 99</td><td colspan="9">Channel number geometry axis for handwheel 1, 2, 3</td></tr>
<tr><td>Edge evaluation: No</td><td colspan="9">Signal(s) updated: Cyclic</td></tr>
<tr><td rowspan="6">Significance of signal</td><td colspan="9">The operator can assign an axis to the handwheel (1, 2, 3) directly on the operator panel front. If this axis is a geometry axis (IS "Machine axis" = 0), the assigned channel number for the handwheel in question is transferred to the PLC.
In this way, the IS "Activate handwheel" is set for the selected geometry axis in accordance with the state set by the operator (IS "Handwheel selected").
The following codes apply to the channel number:</td></tr>
<tr><td colspan="8">Bit</td><td rowspan="2">Channel number</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td colspan="9">With machine axes (IS "Machine axis" = 1), the IS "Channel number geometry axis for handwheel 1, 2, 3" has no meaning.
For further information, see IS "Axis number for handwheel 1, 2, 3".</td></tr>
<tr><td>Corresponding to ....</td><td colspan="9">DB10 DBB100 ff (axis number for handwheel 1, 2, 3)
DB10 DBX100.6 ff (handwheel selected)
DB10 DBX100.7 ff (machine axis)
DB21, ... DBX12.0 - 12.2 ff (activate handwheel)</td></tr>
<tr><td>Application example(s)</td><td colspan="9">If DB10 DBB97 = 2, then handwheel 1 is assigned to channel 2.</td></tr>
</table>

DB10 DBB100, 101, 102 Bit 0 - 4	**Axis number for handwheel 1, 2 or 3**
Edge evaluation: no	Signal(s) updated: Cyclic
Significance of signal	The operator can assign an axis to every handwheel directly via the operator panel front. To do so, he defines the required axis (e.g. X). The basic PLC program provides the number of the axis plus the information "machine axis or geometry axis" (IS "machine axis") as HMI interface signals. The basic PLC program sets the interface signal "Activate handwheel" for the defined axis. Depending on the setting in the HMI interface signal "machine axis", either the interface for the geometry axis or for the machine axis is used. The following must be noted when assigning the axis designation to the axis number: NST "machine axis" = 1; e.g. machine axis: The assignment is done via the machine data: MD10000 $MN_AXCONF_MACHAX_NAME_TAB[n] (machine axis name). NST "machine axis" = 0; e.g. geometry axis: The assignment is done via the machine data: MD20060 $MC_AXCONF_GEOAX_NAME_TAB[n] (geometry axis name in channel) With the NST "Channel number geometry axis handwheel n" the channel number assigned to the handwheel is defined. For following codes are used for the axis number:

Bit					Axis number
4	**3**	**2**	**1**	**0**	
0	0	0	0	0	–
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
0	1	0	0	0	8

Corresponding to	DB10 DBX97 ff (Channel number geometry axis handwheel n) DB10 DBX100.6 ff (handwheel selected) DB10 DBX100.7 ff (machine axis) DB21, ... DBX12.0 to DBX12.2 ff (activate handwheel) DB31, ... DBX4.0 to DBX4.2 (activate handwheel) MD10000 $MN_AXCONF_MACHAX_NAME_TAB [n] (machine axis name) MD20060 $MC_AXCONF_GEOAX_NAME_TAB [n] (geometry axis in the channel)

DB10 DBX100.6, 101.6, 102.6	Handwheel selected (for handwheel 1, 2 or 3)
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The operator has selected the handwheel for the defined axis via the operator panel front (i.e. activated). This information is made available by the basic PLC program at the HMI interface. The basic PLC program sets the interface signal: DB21, ... DBX12.0-12.2 ff (Activate handwheel) for the defined axis to "1". The associated axis is also displayed at the HMI interface: DB10 DBX100.7 ff (machine axis) and DB10 DBB100 ff (axis number for handwheel 1). As soon as the handwheel is active, the axis can be traversed in JOG mode with the handwheel (DB21, ... DBX40.0-40.2 ff (Handwheel active).
Signal state 0 or edge change 1 → 0	The operator has disabled the handwheel for the defined axis via the operator panel front. This information is made available by the basic PLC program at the HMI interface. The basic PLC program can set the interface signal: DB21, ... DBX12.0-12.2 ff (Activate handwheel) for the defined axis to "0".
Corresponding to	DB10 DBB100 ff (axis number) DB10 DBX100.7 ff (machine axis) DB21, ... DBX12.0-12.2 ff (activate handwheel) DB21, ... DBX40.0 - DBX40.2 ff (handwheel active) DB31, ... DBX4.0 - DBX4.2 (activate handwheel) DB10 DBB97 ff (channel number geometry axis for handwheel 1, 2 or 3)

DB10 DBX100.7, 101.7, 102.7	Machine axis (for handwheel 1, 2 or 3)
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The operator has assigned an axis to the handwheel (1, 2, 3) directly on the operator panel front. This axis is a machine axis. For further information see IS "Axis number".
Signal state 0 or edge change 1 → 0	The operator has assigned an axis to the handwheel (1, 2, 3) directly on the operator panel front. This axis is a geometry axis. For further information see IS "Axis number".
Corresponding to	DB10 DBB100 ff (axis number) DB10 DBX100.6 ff (handwheel selected) DB10 DBB97 ff (channel number geometry axis for handwheel 1, 2 or 3)

19.4.2 Signals to channel (DB21, ...)

Overview of signals to channel (to NCK)

DB21, ...	Signals to channel							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0		Activate test run feedrate	Activate M01	Activate single block	Activate DRF	Activate traverse forward	Activate traverse backwards	Work off external
12	Geometry axis 1							
	Traversing keys		Rapid traverse override	Traverse keys disable	Feed stop	Activate handwheel		
	+	-				3	2	1
13	Geometry axis 1 Machine function							
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC
16	Geometry axis 2							
	Traversing keys		Rapid traverse override	Traverse keys disable	Feed stop	Activate handwheel		
	+	-				3	2	1
17	Geometry axis 2 Machine function							
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC
20	Geometry axis 3							
	Traversing keys		Rapid traverse override	Traverse keys disable	Feed stop	Activate handwheel		
	+	-				3	2	1
21	Geometry axis 3 Machine function							
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC

Overview of signals to channel (to NCK)

DB21, ... DBX0.3	**Activate DRF**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The function DRF is selected. The function can either be selected directly from the PLC user program or from the operator panel front via HMI interface signal: DB21, ... DBX24.3 (DRF selected) This HMI interface signal is either converted by the basic PLC program or the PLC user program to interface signal: DB21, … DBX0.3 (Activate DRF). As soon as the function DRF is active, DRF offset can be modified in operating modes AUTOMATIC or MDA.
Signal state 0 or edge change 1 → 0	The function is not selected.
Signal irrelevant for ...	JOG mode
Application example(s)	The DRF function can be enabled specifically by the PLC user program with the interface signal: DB21, … DBX0.3 (Activate DRF).
Corresponding to	DB21, ... DBX24.3 (DRF selected)

DB21, ... DBB12, 16, 20 Bit 0-2	**Activate handwheel (1 to 3) for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	These machine data determine whether this geometry axis is assigned to handwheel 1, 2, 3 or no handwheel. Only one handwheel can be assigned to an axis at any one time. If several interface signals: DB21, … DBX12.1-3, DBX16.1-3, DBX20.1-3 (Activate handwheel) are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies. **Note:** Three geometry axes can be traversed simultaneously with handwheels 1 to 3!
Signal state 0 or edge change 1 → 0	Neither handwheel 1, 2 nor 3 is assigned to this geometry axis.
Application example(s)	The PLC user program can use this interface signal to disable the influence of turning the handwheel on the geometry axis.
Corresponding to	DB21, ... DBX40.7 or DBX40.6 ff (Handwheel active for geometry axis)

DB21, ... DBX12.4, 16.4, 20.4	Traverse key disable for geometry axis (1, 2, 3)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The traverse keys plus and minus have no effect on the geometry axes in question. It is thus not possible to traverse the geometry axis in JOG with the traverse keys on the machine control panel. If the traverse key disable is activated during a traverse movement, the geometry axis is stopped.
Signal state 0	The plus and minus traverse keys are enabled.
Application example(s)	It is thus possible, depending on the operating mode, to disable manual traverse of the geometry axis in JOG mode with the traverse keys from the PLC user program.
Corresponding to	DB21, ... DBX12.7 bzw DBX12.6 ff (Traverse key plus and/or traverse key minus for geometry axis)

DB21, ... DBX12.5, 16.5, 20.5	Input override for geometry axis (1, 2, 3)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	If together with interface signal: DB21, ... DBX12.7 or DBX12.6 ff (traverse key plus or traverse key minus) the PLC interface signal: DB21, … DBB12.5, 16.5 or. 20.5 (Rapid traverse override on geometry axes) is sent, then the corresponding geometry axis is operating with rapid traverse. The rapid traverse feedrate is defined in machine data: MD32010 $MA_JOG_VELO_RAPID (Conventional rapid traverse). The rapid traverse override is effective in the JOG mode for the following versions: • Continuous move • Incremental travel If rapid traverse override is active, the velocity can be modified with the rapid traverse override switch.
Signal state 0 or edge change 1 → 0	The geometry axis traverses with the defined JOG velocity: SD41110 $SN_JOG_SET_VELO (Axis velocity with JOG) or MD32020 $MA_JOG_VELO (Conventional axis velocity).
Signal irrelevant for ...	• Operating modes AUTOMATIC and MDA • Reference point approach (JOG mode)
Corresponding to	DB21, ... DBX12.7 and/or DBX12.6 ff (Traverse key plus and traverse key minus for geometry axis)
References	/FB1/ Function Manual Basic Functions; Feedrates (V1)

DB21, ... DBB12, 16, 20 Bit 7, 6	**Plus and minus traverse keys for geometry axis (1, 2, 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The selected geometry axis can be traversed in both directions in JOG mode with the traverse keys plus and minus. Depending on the active machine function as well as the setting "Jog or continuous mode": SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD (Jog/cont. mode continous with JOG) and MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD (INC and REF in jog mode) different reactions are triggered on a signal change. **Case 1: Continuous traversal in jog mode** The geometry axis traverses in the direction concerned as long as the interface signal is set to 1 (and as long as the axis position has not reached an activated limitation). **Case 2: Continuous traversal in continuous mode** On the first edge change 0 →1 the geometry axis starts to traverse in the relevant direction. This traversing movement still continues when the edge changes from 1 → 0. Any new edge change 0 →1 (same traversing direction!) stops the traversing movement. **Case 3: Incremental traversal in jog mode** With signal 1 the geometry axis starts to traverse at the set increment. If the signal changes to the 0 state before the increment is traversed, the traversing movement is interrupted. When the signal state changes to 1 again, the movement is continued. The geometry axis can be stopped and started several times as described above until it has traversed the complete increment. **Case 4: Incremental traversal in continuous mode** On the first edge change 0 → 1 the geometry axis starts to traverse at the set increment. If another edge change 0 → 1 is performed with the same traverse signal before the geometry axis has traversed the increment, the traversing movement will be cancelled. The increment traversing will then not be completed. If both traverse signals (plus and minus) are set at the same time, no movement occurs, or any current movement is aborted! The effect of the traverse keys can be disabled for every geometry axis individually with the PLC interface signal "Traverse key disable". **Notice!** In contrast to machine axes, for geometry axes, only one geometry axis can be traversed at any one time using the traversing keys. Alarm 20062 is triggered if an attempt is made to traverse more than one axis with the traverse keys.
Signal state 0 or edge change 1 → 0	See cases 1 to 4 above.
Signal irrelevant for ...	Operating modes AUTOMATIC and MDA
Special cases, errors, ...	The geometry axis cannot be traversed in JOG mode: • if it is already being traversed via the axial PLC interface (as a machine axis). • If another geometry axis is already being traversed with the traverse keys. Alarm 20062 "Axis already active" is output.
Corresponding to	DB31, ... DBX8.7 or DBX8.6 (Traverse keys plus and minus for machine axes) DB21, ... DBX12.4 ff (Traverse key disable for geometry axes)

DB21, ... DBB13, 17, 21 Bit 0-5	Machine functions for geometry axis (1, 2, 3) INC1, INC10, INC100, INC1000, INC10000, INCvar
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	This interface signal defines how many increments the geometry axis traverses when the traverse key is pressed or the handwheel is turned one detent position. JOG mode must be active for this (exception: with DRF). The increment size is assigned to the interface signals as follows: • With INC1 to INC10000 With the general machine data: MD11330 $MN_JOG_INCR_SIZE_TAB (increment size for INC/handwheel) • For INCvar With the general setting data: SD41010 $SN_JOG_VAR_INCR_SIZE (size of variable increment in JOG) As soon as the selected machine function becomes active, this is signaled to the PLC interface: DB21, ... DBB41 ff (active machine function INC1; ...) If several machine function signals (INC1, INC... or interface signal "Continuous jogging") are selected at the interface simultaneously, no machine function is activated by the control.
Signal state 0 or edge change 1 → 0	The machine function in question is not selected. If an axis is currently traversing an increment, this movement is also aborted if this machine function is deselected or switched over.
Corresponding to	DB21, ... DBB41 ff (active machine function INC1,...) for geometry axes DB21, ... DBX13 ff (machine function continuous) for geometry axes

DB21, ... DBX13.6, 17.6, 21.6	Machine function continuous for geometry axis (1, 2, 3)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine function "Continuous jogging" is selected. The associated geometry axis can be traversed with the traverse keys plus and minus in JOG mode.
Signal state 0 or edge change 1 → 0	Machine function "Continuous jogging" is not selected.
Corresponding to	DB21, ... DBB41 ff (active machine function INC1, ..., continuous) DB21, ... DBB13 ff (machine function INC1, ..., INC10000)

19.4.3 Signals from channel (DB21, ...)

Overview of signals from channel to PLC

DB21, ...	Signals from channel							
DBB	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
24 (HMI → PLC)		Test run feedrate selected	M01 selected	Single block selected	DRF selected			
33 (HMI → PLC)					Handwheel override active	Activate handwheel		
37	Stop at the end of block with SBL suppressed	Read-in enable is ignored	CLC stopped upper limit /TE1/	CLC stopped lower limit /TE1/	CLC active /TE1/	Contour handwheel active		
						Handwheel 1	Handwheel 2	Handwheel 3
	Geometry axis 1							
40	Travel command		Travel request			Handwheel active		
	plus	minus	plus	minus		3	2	1
41		Geometry axis 1 Active machine function						
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC
	Geometry axis 2							
46	Travel command		Travel request			Handwheel active		
	plus	minus	plus	minus		3	2	1
47		Geometry axis 2 Active machine function						
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC
	Geometry axis 3							
52	Travel command		Travel request			Handwheel active		
	plus	minus	plus	minus		3	2	1
53		Geometry axis 3 Active machine function						
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC

Description of signals from channel to PLC

DB21, ... DBX24.3	DRF selected
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The operator has selected DRF on the operator panel front. The PLC program (basic PLC program or user program) transfers this HMI interface signal to interface signal DB21, ... DBB0.3 (activate DRF) once the corresponding logic operation has been performed. As soon as DRF is active, the DRF offset can be changed in AUTOMATIC or MDA mode using the handwheel assigned to the axis.
Signal state 0 or edge change 1 → 0	The operator has not selected DRF on the operator panel front.
Signal irrelevant for ...	JOG mode
Corresponding to	DB21, ... DBX0.3 (Activate DRF)

DB21, ... DBX33.3	Handwheel override active
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The function "Handwheel override in AUTOMATIC mode" is active for the programmed path axes. Handwheel pulses of the 1st geometry axis function as a velocity override over the programmed path feedrate.
Signal state 0 or edge change 1 → 0	The function "Handwheel override in AUTOMATIC mode" is not active for the programmed path axes. An active handwheel override is not active if: • the path axes have reached the target position. • The distance-to-go is deleted by the channel-specific interface signal DB21, ... DBX6.2 (delete distance-to-go). • a RESET is performed.

DB21, ... DBB37 Bit 0-2	**Contour handwheel active (1 to 3)**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	These PLC interface signals report whether this geometry axis is assigned to contour handwheel 1, 2 or 3 or to no contour handwheel. Only one contour handwheel can be assigned to an axis at any one time. If several interface signals: DB21, … DBB30.0-3 (Activate contour handwheel) are set, priority "Contour handwheel 1" before "Contour handwheel 2" before "Contour handwheel 3" applies. If the assignment is active, the geometry axis can be traversed in JOG mode with the contour handwheel or a DRF offset can be generated in AUTOMATIC or MDA modes.
Signal state 0 or edge change 1 → 0	Neither contour handwheel 1, 2 nor 3 is assigned to this geometry axis.

DB21, ... DBB40, 46, 52 Bit 0-2	**Handwheel active (1 to 3) for geometry axis**
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	These PLC interface signals report whether this geometry axis is assigned to handwheel 1, 2 or 3 or to no handwheel. Only one handwheel can be assigned to an axis at any one time. If several interface signals: DB21, … DBX12.1-3; 16.1-3; 20.1-3 (Activate handwheel) are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies. If the assignment is active, the geometry axis can be traversed in JOG mode with the handwheel or a DRF offset can be generated in AUTOMATIC or MDA modes.
Signal state 0 or edge change 1 → 0	Neither handwheel 1, 2 nor 3 is assigned to this geometry axis.
Corresponding to	DB21, ... DBX12.0-12.2 ff (activate handwheel)

DB21, ... DBB40, 46, 52 Bit 5, 4	**Plus and minus traversing request (for geometry axis)**
Edge evaluation: No	Signal(s) updated: Cyclically
The signal is the same as the travel command signal in the earlier version.	
Signal state 1 or edge change 0 → 1	• JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.

DB21, ... DBB40, 46, 52 Bit 5, 4	**Plus and minus traversing request (for geometry axis)**
Signal state 0 or edge change 1 → 0	A travel command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The travel command is reset depending on the current setting "jog or continuous mode" (DB21, … DBB12.6-7; 16.6-7; 20.6-7). While traversing with the handwheel. • REF mode: When the reference point is reached. AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. NST DB21, … DBX25.7 (axes disable) is active.
Corresponding to	DB21, ... DBX40.7 or DBX40.6 DB21, ... DBX46.7 or DBX46.6 DB21, ... DBX52.7 and/or DBX52.6 (Travel command plus and travel command minus)

DB21, ... DBB40, 46, 52 Bit 7, 6	**Plus and minus travel commands (for geometry axis)**
Edge evaluation: No	Signal(s) updated: Cyclically
The signal has the effect as described, if Bit 0 in the machine data: MD17900 $MN_VDI_FUNCTION_MASK (setting for VDI signals) is set to 0. If bit 0 in the MD is set to 1, then the signal changes to 1 only if the axis is actually moving. The interface signal DB21, ... DBX 40, 46, 52 Bit 5, 4 (traversing request plus/minus), which is always output, has the same effect as signal travel command plus/mins when MD17900 bit 0 = 0.	
Signal state 1 or edge change 0 → 1	A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways: • JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.

DB21, ... DBB40, 46, 52 Bit 7, 6	Plus and minus travel commands (for geometry axis)
Signal state 0 or edge change 1 → 0	A travel command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The travel command is reset depending on the current setting "jog or continuous mode" (see DB21, ... DBX12.7 or DBX12.6 ff). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. NST DB21, … DBX25.7 (axes disable) is active.
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). **Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control!
Corresponding to	DB21, ... DBX12.7 and/or DBX12.6 ff (Traverse key plus and traverse key minus for geometry axis) DB21, ... DBX 40, 46, 52 Bit 5 (Traversing request plus/minus)

DB21, ... DBB41, 47, 53 Bit 0-6	Active machine function for geometry axis (1, 2, 3) INC1, ..., continuous
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The PLC interface receives a signal stating which JOG mode machine function is active for the geometry axes. The reaction to actuation of the traverse key or rotation of the handwheel varies, depending on which machine function is active.
Signal state 0 or edge change 1 → 0	The machine function in question is not active.
Corresponding to	DB21, ... DBB13 ff (machine function INC1, ..., continuously for geometry axis)

DB21, ... DBB332, 336, 340 Bit 5, 4	Plus and minus traversing request (for orientation axis)
Edge evaluation: No	Signal(s) updated: Cyclically
The signal is the same as the travel command signal in the earlier version.	
Signal state 1 or edge change 0 → 1	• JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.

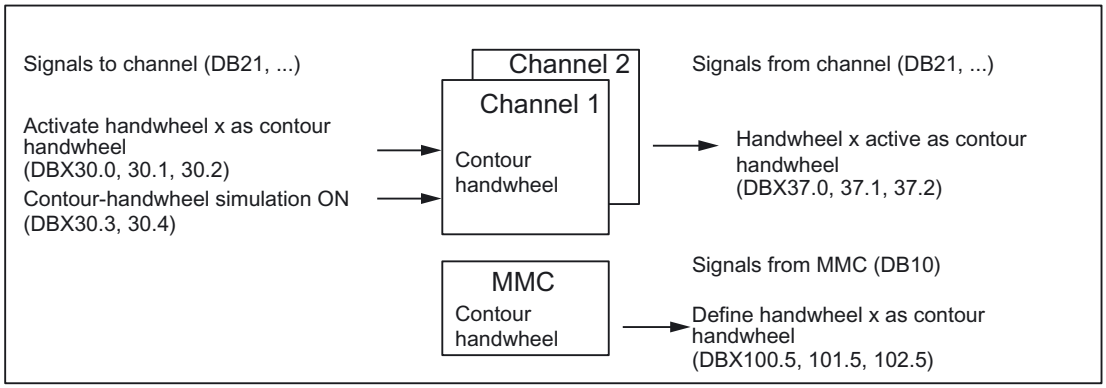
DB21, ... DBB332, 336, 340 Bit 5, 4	Plus and minus traversing request (for orientation axis)
Signal state 0 or edge change 1 → 0	A travel command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The travel command is reset depending on the current setting "jog or continuous mode" (see DBX12.7 and DBX12.6 ff. While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. interface signal DB21, … DBX25.7 (axis disabled) is active.
Corresponding to	DB31, ... DBX332.7 or DBX332.6 DB31, ... DBX336.7 or DBX336.6 DB31, ... DBX340.7 and/or DBX340.6 (traversing command plus and traversing command minus)

DB21, ... DBB332, 336, 340 Bit 7, 6	Plus and minus travel command (for orientation axis)
Edge evaluation: No	Signal(s) updated: Cyclically
The signal has the effect as described, if Bit 0 in the machine data: MD17900 $MN_VDI_FUNCTION_MASK (setting for VDI signals) is set to 0. If bit 0 in the MD is set to 1, then the signal changes to 1 only if the axis is actually moving. The interface signal DB21, ... DBX 332, 336, 340 Bit 5, 4 (traversing request plus/minus), which is always output, has the same effect as signal travel command plus/mins when MD17900 bit 0 = 0.	
Signal state 1 or edge change 0 → 1	A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways: • JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.
Signal state 0 or edge change 1 → 0	A travel command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The travel command is reset depending on the current setting "jog or continuous mode" (see DB21, ... DBX12.7 or DBX12.6 ff). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. interface signal DB21, … DBX25.7 (axes disable) is active.

DB21, ... DBB332, 336, 340 Bit 7, 6	Plus and minus travel command (for orientation axis)
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). Note: If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control!
Corresponding to	DB21, ... DBX12.7 and/or DBX12.6 ff (Traverse key plus and traverse key minus for geometry axis) DB21, ... DBX 332, 336, 340 Bit 5, 4 (Traversing request plus/minus)

19.4.4 Signals with contour handwheel

Overview of interface signals for contour handwheel



Description of interface signals for contour handwheel

DB21, ... DBX30.0 DBX30.1 DBX30.2	Activate handwheel 1 as contour handwheel Activate handwheel 2 as contour handwheel Activate handwheel 3 as contour handwheel	
Edge evaluation: No		Signal(s) updated: Cyclic
Description	One of the three handwheels can be selected/deselected as contour handwheel via these signals:	
	Signal = 1	Handwheel x is selected as contour handwheel
	Signal = 0	Handwheel x is deselected as contour handwheel

DB21, ... DBX30.0 DBX30.1 DBX30.2	**Activate handwheel 1 as contour handwheel** **Activate handwheel 2 as contour handwheel** **Activate handwheel 3 as contour handwheel**
	Enabling/disabling of the contour handwheel can be performed in the middle of a block. Upon enabling, the movement is first decelerated and then traversed according to the contour handwheel. Upon disabling, the movement is decelerated and the NC program is continued immediately. If the NC program is to be continued only after a new NC START, then deactivation of the contour handwheel in the PLC user program must be combined with an NC STOP.
Special cases, errors, ...	The signal is maintained after NC RESET.
Corresponding to	DB21, ... DBX37.0, 37.1, 37.2 (Handwheel x active as contour handwheel)

DB21, ... DBX30.3 DBX30.4	**Simulation contour handwheel on** **Negative direction simulation contour handwheel**		
Edge evaluation: No	Signal(s) updated: Cyclic		
Description	For enabling/disabling simulation of the contour handwheel, and for definition of the traversing direction, these signals have to be set as follows:		
	Bit 3	**Bit 4**	**Meaning**
	0	0	Simulation OFF
	0	1	Simulation OFF
	1	0	Simulation ON, direction as programmed
	1	1	Simulation ON, direction opposite programmed direction
	During simulation the feedrate is no longer defined by the contour handwheel, but traversing occurs with the programmed feedrate on the contour. When the function is deselected, the runniung movement is decelerated by the braking ramp. When the traversing direction is switched, the running movement is decelerated by the braking ramp, and traversing occurs in the opposite direction.		
Special cases, errors, ...	Simulation is only effective in AUTOMATIC mode and can only be activated when the contour handwheel is activated.		

DB21, ... DBX37.0 DBX37.1 DBX37.2	**Handwheel 1 active as contour handwheel** **Handwheel 2 active as contour handwheel** **Handwheel 3 active as contour handwheel**
Edge evaluation: No	Signal(s) updated: Cyclic
Description	These signals show which handwheel is selected as contour handwheel:
	Signal = 1 Handwheel x is selected as contour handwheel.
	Signal = 0 Handwheel x is deselected as contour handwheel.
Special cases, errors, ...	The signal is maintained after NC RESET.
Corresponding to	DB21, ... DBX30.0, 30.1, 30.2 (Handwheel x active as contour handwheel)

DB10 DBX100.5 DBX101.5 DBX102.5	** Define handwheel 1 as contour handwheel Define handwheel 2 as contour handwheel Define handwheel 3 as contour handwheel**
Edge evaluation: No	Signal(s) updated: Cyclic
Description	These signals show which handwheel is defined as contour handwheel via MMC:
	Signal = 1 Handwheel x is defined as contour handwheel via MMC.
	Signal = 0 Handwheel x is not defined as contour handwheel.
	In order for the handwheel defined via MMC to become effective as contour handwheel, the corresponding signal has to be combined on interface signal: DB21, ... DBX30.0, 30.1, 30.2 (Activate handwheel as contour handwheel).
Special cases, errors, ...	Depending on the settings of parameter HWheelMMC in FB1 of the basic PLC program, these signals are either supplied by the basic program or must be supplied by the PLC user program.
Corresponding to	DB21 ... DBX30.0, 30.1, 30.2 (Activate handwheel x as contour handwheel) FB1 parameters HWheelMMC

19.4.5 Signals to axis/spindle (DB31, ...)

Overview of signals to axis/spindle

DB31, ...	**Signals to axis/spindle**							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
4	Traversing keys		Rapid traverse override	Traverse keys disable	Feed stop Spindle stop	Activate handwheel		
	plus	minus				3	2	1
5	Machine function							
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC
13						JOG - Approaching fixed point		

Description of signals to axis/spindle

DB31, ... DBB4 Bit 0-2	Activate handwheel (1 to 3)
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	This PLC interface signal defines whether this machine axis is assigned to handwheel 1, 2, 3 or no handwheel. Only one handwheel can be assigned to an axis at any one time. If several interface signals: DB31, ... DBX4.0, 4.1, 4.2 (Activate handwheel) are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies. If the assignment is active, the machine axis can be traversed with the handwheel in JOG mode or a DRF offset can be generated in AUTOMATIC or MDA mode.
Signal state 0 or edge change 1 → 0	Neither handwheel 1, 2 nor 3 is assigned to this geometry axis.
Application example(s)	The PLC user program can use this interface signal to disable the influence of turning the handwheel on the axis.
Corresponding to	DB31, ... DBX64.0 to DBX64.2 (Handwheel active)

DB31, ... DBX4.4	Traverse key disable
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The traverse keys plus and minus have no effect on the machine axes in question. It is thus not possible to traverse the machine axis in JOG with the traverse keys on the machine control panel. If the traverse key disable is activated during a traverse movement, the machine axis is stopped.
Signal state 0 or edge change 1 → 0	The plus and minus traverse keys are enabled.
Application example(s)	It is thus possible, depending on the operating mode, to disable manual traverse of the machine axis in JOG mode with the traverse keys from the PLC user program.
Corresponding to	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus)

DB31, ... DBX4.5	Rapid traverse override
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	If together with interface signal: DB31, ... DBX4.0, 4.1, 4.2 (traverse key plus or traverse key minus) the PLC interface signal: DB31, ... DBX4.5 (Rapid traverse override) is sent, then the corresponding machine axis is operating with rapid traverse. The rapid traverse feedrate is defined in machine data: MD32010 $MA_JOG_VELO_RAPID (Conventional rapid traverse). The rapid traverse override is effective in the JOG mode for the following versions: • Continuous jogging • Incremental jogging If rapid traverse override is active, the velocity can be modified with the rapid traverse override switch.
Signal state 0 or edge change 1 → 0	The machine axis traverses with the defined JOG velocity: SD41110 $SN_JOG_SET_VELO (Axis velocity with JOG) or MD32020 $MA_JOG_VELO (Conventional axis velocity).
Signal irrelevant for ...	Operating modes AUTOMATIC and MDA Reference point approach (JOG mode)
Corresponding to	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus) DB31, ... DBB0 (axial feed/spindle override)

DB31, ... DBB4 Bit 7, 6	Plus and minus traverse keys
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The selected machine axis can be traversed in both directions in JOG mode with the traverse keys plus and minus. Depending on the active machine function as well as the setting of the setting data: SD41050 $SN_JOG_CONT_MODE_LEVELTRIGGRD (Jog/cont. mode continous with JOG) continuous with JOG and/or in machine data: MD11300 $MN_JOG_INC_MODE_LEVELTRIGGRD when JOG-INCR (INC and REF in jog mode), different reactions are triggered on a signal change. **Case 1: Continuous traversal in jog mode** The machine axis traverses in the direction concerned as long as the interface signal is set to 1 (and as long as the axis position has not reached an activated limitation). **Case 2:Continuous traversal in continuous mode** On the first edge change 0 →1 the machine axis starts to traverse in the relevant direction. This traversing movement still continues when the edge changes from 1 → 0. Any new edge change 0 →1 (same traversing direction!) stops the traversing movement. **Case 3:Incremental traversal in jog mode** With signal 1 the machine axis starts to traverse at the set increment. If the signal changes to the 0 state before the increment is traversed, the traversing movement is interrupted. When the signal state changes to 1 again, the movement is continued. The axis can be stopped and started several times as described above until it has traversed the complete increment. **Case 4: Incremental traversal in continuous mode** On the first edge change 0 → 1 the machine axis starts to traverse at the set increment. If another edge change 0 → 1 is performed with the same traverse signal before the axis has traversed the increment, the traversing movement will be cancelled. The increment traversing will then not be completed. If both traverse signals (plus and minus) are set at the same time there is no movement or a current movement is aborted. The effect of the traverse keys can be disabled for every machine axis individually with the PLC interface signal: DB31, ... DBX4.4 (Traverse key disable).
Signal state 0 or edge change 1 → 0	See cases 1 to 4 above.
Signal irrelevant for ...	Operating modes AUTOMATIC and MDA
Application example(s)	The machine axis cannot be traversed in JOG mode if it is already being traversed via the channel-specific PLC interface (as a geometry axis). Alarm 20062 is signaled.
Special cases, ...	Indexing axes
Corresponding to	DB21, ... DBX12.7, DBX12.6 ff (Traverse keys plus and minus for geometry axes) DB31, ... DBX4.4 (traversing key disable)

DB31, ... DBB5 Bit 0-5	Machine function INC1, INC10, INC100, INC1000, INC10000, INCvar
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	This interface signal defines how many increments the machine axis traverses when the traverse key is pressed or the handwheel is turned one detent position. JOG mode must be active for this (exception: with DRF). The increment size associated with the individual interface signals is defined as follows: • For INC1 to INC10000: With the general machine data: MD11330 $MN_JOG_INCR_SIZE_TAB (increment size for INC/handwheel) • For INCvar: With the general setting data: SD41010 $SN_JOG_VAR_INCR_SIZE (size of variable increment in JOG) As soon as the selected machine function becomes active, this is signaled to the PLC interface: DB31, ... DBB65 (active machine function INC1; ...) If several machine function signals (INC1, INC... or DB31, ... DBX5.6) are selected at the interface simultaneously, no machine function is activated by the control.
Signal state 0 or edge change 1 → 0	The machine function in question is not selected. If an axis is currently traversing an increment, this movement is also aborted if this machine function is deselected or switched over.
Corresponding to	DB31, ... DBB65 (machine function INC1, ...) DB31, ... DBX5.6 (Machine function continuous)

DB31, ... DBX5.6	Continuous machine function
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	The machine function: DB31, ... DBX5.6 (continuous jogging) is selected. The associated machine axis can be traversed with the traverse keys plus and minus in JOG mode.
Signal state 0 or edge change 1 → 0	The machine function: DB31, ... DBX5.6 (continuous jogging) is not selected.
Corresponding to	DB31, ... DBB65 (active machine function INC1, ..., continuous) DB31, ... DBB5 (machine function INC1, ..., INC10000)

DB31, ... DBB13 Bit 0-2	JOG - Approaching fixed point 0/1/2
Edge evaluation: No	Signal(s) updated: Cyclically
Signal state 1 or edge change 0 → 1	Activates the "approaching fixed point JOG" function. The number of the fixed point to be approached is output through Bit 0-2 binary code. The selected machine axis can be traversed to the corresponding fixed point with the traverse keys or the handwheel as soon as the function is active (see DB31, ... DBX75.0-2). The available fixed points (max. 4 for each axis) are defined via the following machine data: MD30600 $MA_FIX_POINT_POS[n]
Signal state 0 or edge change 1 → 0	Deactivates the "approaching fixed point JOG" function.
Corresponding to	DB31, ... DBX75.0-2 (JOG - Approach fixed point) DB31, ... DBX75.3-5 (JOG - Approach fixed point) MD30600 $MA_FIX_POINT_POS[n] (fixed value positions of the axis)

19.4.6 Signals from axis/spindle (DB31, ...)

Overview of signals from axis/spindle

DB31, ...	Signals from axis/spindle							
DBB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
64	Travel command		Travel request			Handwheel active		
	plus	minus	plus	minus		3	2	1
65		Active machine function						
		continuous-ly	Variable INC	10000 INC	1000 INC	100 INC	10 INC	1 INC
75			JOG - Approaching fixed point reached			JOG - Approaching fixed point active		

Description of signals from axis/spindle

DB31, ... DBX62.1	**Handwheel override active**
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The function "Handwheel override in AUTOMATIC mode" is active for the programmed positioning axis (FDA[AXi]). Handwheel pulses for this axis affect the programmed axis feedrate either as path definition (FDA=0) or as velocity override (FDA > 0). The interface signal will also be set if "Handwheel override in automatic mode" is active with a concurrent positioning axis with FC18 (for 840D sl).
Signal state 0 or edge change 1 → 0	The function "Handwheel override in AUTOMATIC mode" is not active for the programmed positioning axis (or concurrent positioning axis). An active handwheel override is not active if: • the positioning axis has reached the target position • the distance-to-go is deleted by axis-specific interface signal DB31, ... DBX2.2 (delete distance-to-go). • a RESET is performed.

DB31, ... DBB64 Bit 0-2	**Handwheel active (1 to 3)**
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	These PLC interface signals provide feedback whether the machine axis is assigned to handwheel 1, 2, 3 or no handwheel. Only one handwheel can be assigned to an axis at any one time. If several interface signals: DB31, ... DBX4.0 to DBX4.2 (Activate handwheel) are set, priority "Handwheel 1" before "Handwheel 2" before "Handwheel 3" applies. If the assignment is active, the machine axis can be traversed with the handwheel in JOG mode or a DRF offset can be generated in AUTOMATIC or MDA mode.
Signal state 0 or edge change 1 → 0	Neither handwheel 1, 2 nor 3 is assigned to this geometry axis.
Corresponding to	DB31, ... DBX4.0 to DBX4.2 (activate handwheel) DB10 DBB100.6 ff (handwheel selected)

DB31, ... DBB64 Bit 5, 4	**Plus and minus traversing request**
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways: • JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.
Signal state 0 or edge change 1 → 0	A travel command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The travel command is reset depending on the current setting "jog or continuous mode" (see interface signal DB31, ... DBX4.7 or DBX4.6). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. DB21, … DBX25.7 (axes disable) is active.
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). **Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control!
Corresponding to	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus) DB31, ... DBX64.7 and/or DBX64.6 (Travel command plus and minus)

DB31, ... DBB64 Bit 7, 6	**Plus and minus traverse keys**
Edge evaluation: no	Signal(s) updated: Cyclic
The signal has the effect as described, if Bit 0 in the machine data: MD17900 $MN_VDI_FUNCTION_MASK (setting for VDI signals) is set to 0. If bit 0 in the MD is set to 1, then the signal changes to 1 only if the axis is actually moving. The interface signal DB31, ... DBX64 Bit 5, 4 (traversing request plus/minus), which is always output, has the same effect as signal travel command plus/minus when MD17900 bit 0 = 0.	

DB31, ... DBB64 Bit 7, 6	Plus and minus traverse keys
Signal state 1 or edge change 0 → 1	A traverse movement of the axis is to be executed in one or the other direction. Depending on the mode selected, the command is triggered in different ways: • JOG mode: With the plus or minus traverse key. • REF mode: With the traverse key that takes the axis to the reference point. • AUT/MDA mode: A program block containing a coordinate value for the axis in question is executed.
Signal state 0 or edge change 1 → 0	A travel command in the relevant axis direction has not been given or a traverse movement has been completed. • JOG mode: The travel command is reset depending on the current setting "jog or continuous mode" (DB31, ... DBX4.7 and/or DBX4.6). While traversing with the handwheel. • REF mode: When the reference point is reached. • AUT/MDA mode: The program block has been executed (and the next block does not contain any coordinate values for the axis in question). Cancel by "RESET", etc. DB21, … DBX25.7 (axes disable) is active.
Application example(s)	To release clamping of axes with clamping (e.g. on a rotary table). **Note:** If the clamping is not released until the travel command is given, these axes cannot be operated under continuous path control!
Corresponding to	DB31, ... DBX4.7 and/or DBX4.6 (traversing key plus and traversing key minus) DB31, ... DBX64.5 and/or DBX.4 (Traversing request plus and minus)

DB31, ... DBB65 Bit 0-6	Active machine function INC1, ..., continuous
Edge evaluation: no	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The PLC interface receives a signal stating which JOG mode machine function is active for the machine axes. The reaction to actuation of the traverse key or rotation of the handwheel varies, depending on which machine function is active.
Signal state 0 or edge change 1 → 0	The machine function in question is not active.
Corresponding to	DB31, ... DBB5 (machine function INC1, ..., continuous)

DB31, ... DBB75 Bit 0-2	**JOG - Approaching fixed point active**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Message to the PLC that the function "Approaching fixed point in JOG" is effective. The selected machine axis can be traversed to the specified fixed point binary-coded via Bit 0-2 with the traverse keys or the handwheel.
Signal state 0 or edge change 1 → 0	"Approaching fixed point in JOG" is not active
Corresponding to	DB31, ... DBX13.0-2 (JOG - Approach fixed point) DB31, ... DBX75.3-5 (JOG - Approach fixed point)

DB31, ... DBB75 Bit 3-5	**JOG - Approaching fixed point reached**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Message to PLC that the selected axis has reached the approaching fixed point with "exact stop fine" by virtue of the traversing motion in JOG. This display signal is also signaled if the axis reaches the fixed point position in the machine coordinates system via other methods e.g. NC program, FC18 (for 840D sl) or synchronized action on the setpoint side and comes to a standstill on the actual value side within the "Exact stop fine" tolerance window (MD36010 $MA_STOP_LIMIT_FINE).
Signal state 0 or edge change 1 → 0	The axis has not yet reached the approaching fixed point.
Corresponding to	DB31, ... DBX13.0-2 (JOG - Approach fixed point) DB31, ... DBX75.0-2 (JOG - Approach fixed point)

19.5 Compensations (K3)

No signal descriptions required.

19.6 Mode groups, channels, axis replacement

19.6.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBB8	**Axis/spindle replacement**	
Edge evaluation: Yes		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The current axis type and currently active channel for this axis must be specified. With axis replacement by the PLC, the bit meanings of the signal to axis/spindle DB31, ... DBB8:	
	Bit 0:	Assign A NC axis/spindle channel
	Bit 1:	B ...
	Bit 2:	C ....
	Bit 3:	Assign D NC axis/spindle channel
	Bit 4:	Activation, assignment by means of a positive edge
	Bit 5:	-
	Bit 6:	-
	Bit 7:	Request PLC axis/spindle
Signal state 0 or edge change 1 → 0		
Corresponding to	DB31, ... DBB68 (Axis/spindle replacement) MD20070 $MC_AXCONF_ASSIGN_MASTER_USED (Machine axis number valid in channel) MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN (Initial setting of channel for axis replacement)	
Special cases, errors, ...		

19.6.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBB68	**Axis/spindle replacement**	
Edge evaluation: Yes		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The current axis type and currently active channel for this axis is displayed. With axis replacement by the PLC, the bit meanings of the signal from axis/spindle DB31, ... DBB68:	
	Bit 0:	A NC axis/spindle in channel
	Bit 1:	B ...
	Bit 2:	C ....
	Bit 3:	D NC axis/spindle in channel
	Bit 4:	New type requested from PLC
	Bit 5:	Axis replacement possible
	Bit 6:	neutral axis/spindle as well as command/oscillation axes

DB31, ... DBB68	Axis/spindle replacement	
	Bit 7:	PLC axis/spindle
Signal state 0 or edge change 1 → 0		
Corresponding to	DB31, ... DBB8 (Axis/spindle replacement) MD20070 $MC_AXCONF_ASSIGN_MASTER_USED (Machine axis number valid in channel) MD30550 $MA_AXCONF_ASSIGN_MASTER_CHAN (Initial setting of channel for axis replacement)	
Special cases, errors, ...		

19.7 Kinematic transformation

19.7.1 Signals from channel (DB21, ...)

DB21, ... DBX33.6	Transformation active	
Edge evaluation: No		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The NC command TRANSMIT, TRACYL, TRAANG or TRAORI is programmed in the part program. The corresponding block has been processed by the NC and a transformation is now active.	
Signal state 0 or edge change 1 → 0	No transformation active.	
References	/PGA/ Programming Manual Job Planning /FB3/ Function Manual, Special Functions; 3- to 5-Axis Transformation (F2).	

19.8 Measurement

19.8.1 Signals from NC (DB10)

DB10 DBX107.0 and DBX107.1	Probe actuated
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Probe 1 or 2 is actuated.
Signal state 0 or edge change 1 → 0	Probe 1 or 2 is not actuated.
References	/PHD/ Equipment Manual NCU

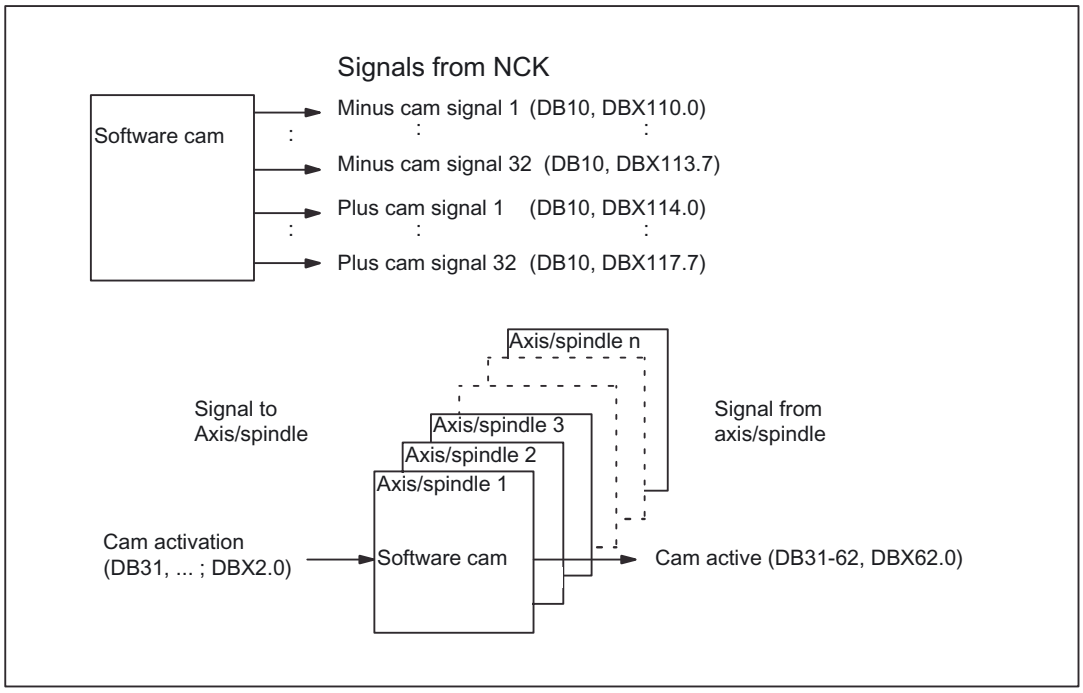
19.8.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX62.3	Measuring status
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The "Measuring" function is active. This signal is used during measuring and displays the current measuring status of the axis.
Signal state 0 or edge change 1 → 0	The "Measuring" function is not active.

19.9 Software cams, position switching signals

19.9.1 Signal overview

PLC interface signals for "Software cams, position switching signals"



19.9.2 Signals from NC (DB10)

DB10 DBX110.0-113.7	**Minus cam signal 1-32**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The switching edge of the minus cam signal 1-32 is generated as a function of the traversing direction of the (rotary) axis and transferred to the PLC interface in the IPO cycle. **Linear axis:** The minus cam signal switches from 0 to 1 if the axis overtravels the minus cam in the negative axis direction. **Modulo rotary axis:** The minus cam signal changes level in response to every positive edge of the plus cam signal.
Signal state 0 or edge change 1 → 0	**Linear axis:** The minus cam signal switches from 1 to 0 when the axis traverses the minus cam in the positive axis direction. **Modulo rotary axis:** The minus cam signal changes level in response to every positive edge of the plus cam signal.

DB10 DBX114.0-117.7	**Plus cam signal 1-32**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The switching edge of the plus cam signal 1-32 is generated as a function of the traversing direction of the (rotary) axis and transferred to the PLC interface in the IPO cycle. **Linear axis:** The plus cam signal switches from 0 to 1 when the axis traverses the plus cam in the positive direction. **Modulo rotary axis:** The plus cam signal switches from 0 to 1 when the minus cam is overtraveled in the positive axis direction. The described response of the plus cam applies under the condition: plus cam - minus cam < 180 degrees If this condition is not fulfilled or if the minus cam is set to a greater value than the plus cam, then the response of the plus cam signal is inverted. The response of the minus cam signal remains unchanged.
Signal state 0 or edge change 1 → 0	**Linear axis:** The plus cam signal switches from 1 to 0 if the axis overtravels the plus cam in the negative direction. **Modulo rotary axis:** The plus cam signal switches from 1 back to 0 if the plus cam is overtraveled in the positive axis direction. The described response of the plus cam applies under the condition: plus cam - minus cam < 180 degrees If this condition is not fulfilled or if the minus cam is set to a greater value than the plus cam, then the response of the plus cam signal is inverted. The response of the minus cam signal remains unchanged.

19.9.3 Signals to axis/spindle (DB31, ...)

DB31, ... DBX2.0	Cam activation	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Output of the minus and plus cam signals of an axis to the general PLC interface is activated. The activation takes effect immediately after processing of IS "Cam activation".	
Signal state 0 or edge change 1 → 0	The minus and plus cam signals of an axis are **not** output to the general PLC interface.	
Corresponding to	DB10 DBX110.0 - 113.7 (minus cam signal 1-32) DB10 DBX114.0 - 117.7 (plus cam signals 1-32)	

19.9.4 Signals from axis/spindle (DB31, ...)

DB31, ... DBX62.0	Cams active	
Edge evaluation: no		Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	All cams of the axis selected via NC/PLC interface signal: DB31, ... DBX2.0 (Cam activation) have been activated successfully.	
Signal state 0 or edge change 1 → 0	The cams of the axis are **not** activated.	
Corresponding to	DB31, ... DBX2.0 (Cam activation) DB10 DBX110.0 - 113.7 (minus cam signal 1-32) DB10 DBX114.0 - 117.7 (plus cam signals 1-32)	

19.10 Punching and nibbling

19.10.1 Signal overview

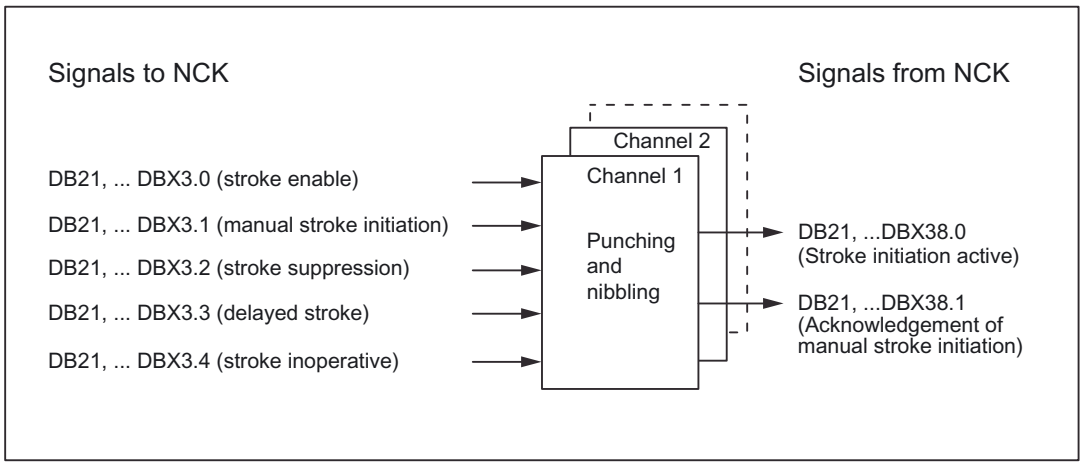


Figure 19-1 PLC interface signals for "Punching and nibbling"

19.10.2 Signals to channel (DB21, ...)

DB21, ... DBX3.0	**No stroke enable**
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal releases the punching strokes via the PLC. 1 signal: The stroke is locked, the NC may not trigger a punching stroke.
Signal state 0 or edge change 1 → 0	0 signal: Punching stroke is available. As long as release is not set, the NC may perform a punching stroke

DB21, ... DBX3.1	Manual stroke initiation
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal enables the triggering of a single stroke in manual mode. 1 signal: Manual stroke is performed.
Signal state 0 or edge change 1 → 0	0 signal: No effect.

DB21, ... DBX3.2	Stroke suppression
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal simply prevents execution of the stroke. The machine traverses anyway. The automatic path segmentation remains active if it is already activated. Only the signal "Stroke initiation" is suppressed. The machine traverses in "stop and go" mode. The step length is defined via the path segmentation. 1 signal: Stroke suppression is active.
Signal state 0 or edge change 1 → 0	0 signal: Stroke suppression is not active.

DB21, ... DBX3.3	Delayed stroke
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	A "Delayed stroke" can be activated via this signal. This corresponds in function to the programming of `PDELAYON`. Other PLC signals not corresponding to the standard are not evaluated in the NCK. With the exception of the manual stroke initiation, the evaluation of signals is limited to `PON` active. 1 signal: Delayed stroke is active.
Signal state 0 or edge change 1 → 0	0 signal: Delayed stroke is not active.

DB21, ... DBX3.4	Stroke inoperative
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The NC reacts to this signal by initiating an immediate movement stop. An alarm is output if any other movement or action needs to be interrupted as a result of this signal. In physical terms, the signal is identical to the signal "Stroke active" for the CNC, i.e. the system is wired in such a way that the two signals are taken to the same NC input via an AND gate. 1 signal: Stroke inoperative (corresponds to the signal "stroke enable").
Signal state 0 or edge change 1 → 0	0 signal: Stroke operative (corresponds to the signal "stroke enable").

DB21, ... DBX3.5	Manual stroke initiation
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The signal "manual stroke initiation" allows the operator to initiate a punching process, even when the parts program is not being processed. Thus the initiation of the punching process is controlled from the PLC. Successful stroke initiation is indicated to the PLC by the NCK-PLC interface signal: DB21, ... DBX38.1 (Manual stroke initiation acknowledgement). 1 signal: Manual stroke initiation is active.
Signal state 0 or edge change 1 → 0	0 signal: Manual stroke initiation is not active.

19.10.3 Signals from channel (DB21, ...)

DB21, ... DBX38.0	Stroke initiation active
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal displays whether the stroke initiation is active. 1 signal: Stroke initiation is active.
Signal state 0 or edge change 1 → 0	0 signal: Stroke initiation is not active.

DB21, ... DBX38.1	**Acknowledgement of manual stroke initiation**
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	This signal displays whether a manual stroke has been initiated. 1 signal: Manual stroke has been performed.
Signal state 0 or edge change 1 → 0	0 signal: Manual stroke has not been performed.

19.11 Positioning axes

The following signals or commands on the NCK-HMI-PLC interface are only of significance for the positioning axis:

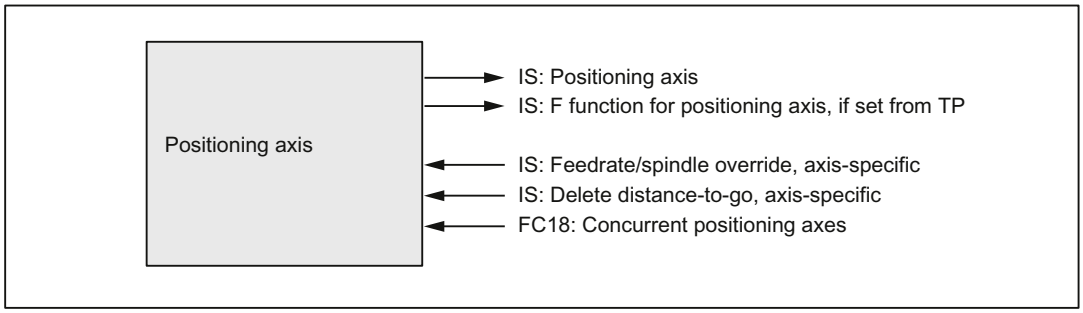


Figure 19-2 Signal modification by the PLC

19.11.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBB0	**Feedrate override/spindle override axis-specific**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Positioning axes have their own axis-specific feed override value. This feedrate override is evaluated in the same way as the channel-specific feedrate override.
Signal irrelevant for ...	NST DB31, ... DBX74.5 ("Positioning axis") = ZERO
References	Evaluation see: DB21, ... DBB4 (feed rate override); channel-specific

DB31, ... DBX2.2	**Delete distance-to-go, axis-specific**
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis-specific distance-to-go of the positioning axis is canceled. The positioning axis is decelerated and the following error is eliminated. The programmed end position is deemed to have been reached. The path axes are not influenced by the axis-specific "delete distance-to-go" interface signal. The channel-specific "delete distance-to-go" interface signal is used for this purpose.
Special cases, errors, ...	If the axis-specific "delete distance-to-go" interface signal is enabled, even if no positioning axes have been programmed in this block, the NCK does not respond.
Corresponding to	DB21, ... DBX6.2 (delete distance-to-go); channel-specific for path axes

DB31, ... DBX28.1	AxReset Reset axis/spindle
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	An axis/spindle controlled from the PLC is to be reset again. The NCK switches the active axis/spindle to the stop state with IS DB31 ... DBX63.2 ("AXSTOP active").
Special cases, errors, ...	Secondary condition: The axis/spindle must have actually accepted the PLC and be controlled by the PLC.
Corresponding to	DB31, ... DBX63.2 (AXRSTOP); Stop axis/spindle

DB31, ... DBX28.2	AxResume Resume axis/spindle motion
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	An interpolating axis/spindle from the main run continues independently of the NC program and is controlled by the PLC. The NCK checks whether an axis/spindle axial alarm with the cancel criterion CANCELCLEAR or NCSTARTCLEAR is active and deletes it. The axis-specific RESUME (DB31 ... DBX28.2) instruction can be aborted by the NCK with IS DB31 … DBX28.2 ("AXSTOP active").
Special cases, errors, ...	Secondary condition: The axis/spindle must be operating under PLC control. The IS DB31 … DBX28.2 ("AXRESUME") is ignored in the following error states detected by the NCK: When the PLC prompts axes/spindles to continue traversing and this axis/spindle has not been accepted by the PLC. The axis/spindle is not in the stopped state. The axis/spindle cannot resume traversal because an alarm is active.
Corresponding to	DB31, … DBX28.1 (AXRESET); Reset axis/spindle

DB31, ... DBX61.1	AxAlarm Axial stop alarm for this axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The NCK brakes the axis/spindle along a ramp and confirms the braking operation via the OPI. At the same time the PLC alarm is sent with IS DB31, ... DBX61.1 ("Axial alarm") == 1 and the status system variable $AA_SNGLAX_STAT == 5 is set.
Corresponding to	System variable $AA_SNGLAX_STAT == 5

DB31, ... DBX63.0	AxReset done Axial reset has been performed
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	After the axial machine data activated by Reset have become operative, the IS DB31 ... DBX63.0 ("AXRESET DONE") == 1, IS DB31, ... DBX63.2 ("Axis stop active") == 0 and system variable $AA_SNGLAX_STAT == 1 single axis in RESET is reported active.
Corresponding to	DB31 ... DBX63.2 ("Axis stop active") == 0 System variable $AA_SNGLAX_STAT == 1

DB31, ... DBX63.1	PLC-controlled axis Axis control passed to PLC
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis/spindle takeover confirmed by the NCK is transferred with the axis/spindle status by the IS DB31, ... DBX63.1 ("PLC-controlled axis") == 1 to the PLC. The current axis status can be checked with system variable $AA_SNGLAX_STAT.
Corresponding to	DB31 … DBX28.7 (PLC-controlled axis); axis is controlled by PLC. System variable $AA_SNGLAX_STAT current axis status

DB31, … DBX63.2	AxStop active Acknowledgement of stopped status
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	NCK puts the axis/spindle with the IS DB31, ... DBX63.2 ("Axis stop active") == 0 into the status stopped. All axis motions controlled by the main run are stopped, Axis status interrupted with system variable $AA_SNGLAX_STAT == 3
Corresponding to	DB31 … DBX28.6 (AXSTOP, Stop); Stop axis/spindle System variable $AA_SNGLAX_STAT == 3

DB31, ... DBX76.5	Positioning axis NCK treats the axis as a positioning axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The NCK treats the axis as a positioning axis. By this, it has: • its own axis interpolator (linear interpolator) • its own feedrate (F value) • its own feedrate override • Exact stop (G09) at the progr. end position

DB31, ... DBB78-81	F function (feedrate) for positioning axis Assign feedrate using the programmed axis name of a positioning axis
Edge evaluation: No	Signal(s) updated: when change
Signal state 1 or edge change 0 → 1	The programmed axial feed is assigned to a positioning axis by means of the programmed axis name and output to the PLC for this axis. The value specified by FC18 (for 840D sl is not output.
Signal irrelevant for ...	IS: Positioning axis = ZERO
Special cases, errors, ...	If the positioning axis is traversed with the feedrate from the following machine data, the NC does not output an F function (feed) to the PLC: MD32060 $MA_POS_AX_VELO (initial setting for positioning axis velocity)
Corresponding to	DB31, ... DBX74.5 (Positioning axis) MD22240 $MC_AUXFU_F_SYNC_TYPE (output time of F functions)

19.11.2 Function call - only 840D sl

FC18

For SINUMERIK 840D sl, concurrent positioning axes can be started from the PLC using FC18 (Function Call 18) of the PLC. The following parameters are passed to the function call:

- Axis name/axis number
- End position
- Feedrate

  (for feedrate = 0, the feedrate is taken from MD32060 $MA_POS_AX_VELO)

  The F value of FC18 is **not** transferred to the axis-specific IS DB31, ...DBB78-81 ("F function (feedrate) for positioning axis")
- Absolute coordinates (G90), incremental coordinates (G91), absolute coordinates along the shortest path for rotary axes (rotary axis name = DC(value))

Since each axis is assigned to exactly one channel, the control can select the correct channel from the axis name/axis number and start the concurrent positioning axis on this channel.

Reference:
Function Manual Basic Functions; PLC Basic Program for SINUMERIK 840D sl (P3)

19.12 Oscillation

19.12.1 Signals to axis/spindle (DB31, ...)

VDI input signals

The PLC user program uses the following signals to control the oscillation process.

DB31, ... DBX28.0	External oscillation reversal
Edge evaluation: Yes	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Brake oscillation motion and move oscillation axis in the opposite direction.
Signal state 0 or edge change 1 → 0	Continue oscillation without interruption

DB31, ... DBX28.3	Set reversal point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Reversal point 2
Signal state 0 or edge change 1 → 0	Reversal point 1

DB31, … DBX28.4	Alter reversal point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The selected reversal point can be altered by manual traverse. In conjunction with DB31, ...DBX28.0: The position at which axis is braked after external oscillation reversal must be accepted as new reversal point.
Signal state 0 or edge change 1 → 0	The selected reversal point cannot be altered by manual traverse. In conjunction with DB31, ...DBX28.0: No change to reversal point
Corresponding to	DBX28.3

DB31, ... DBX28.5	Stop at next reversal point
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The oscillation movement is interrupted at the next reversal point.
Signal state 0 or edge change 1 → 0	The oscillation movement continues after the next reversal point.
Corresponding to	DBX28.6, DBX28.7

DB31, ... DBX28.6	Stop along braking ramp
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is decelerated along a ramp, the oscillation movement is interrupted.
Signal state 0 or edge change 1 → 0	The oscillation movement continues without interruption.
Corresponding to	DBX28.5, DBX28.7

DB31, ... DBX28.7	PLC controls axis
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Axis is controlled by the PLC. The reaction to interface signals is controlled by the PLC by means of the 2 stop bits, other signals with deceleration action are ignored.
Signal state 0 or edge change 1 → 0	Axis is not controlled by the PLC.
Corresponding to	DBX28.5, DBX28.6

19.12.2 Signals from axis/spindle (DB31, ...)

VDI output signals

The NCK makes the following signals available to the PLC user program.

DB31, ... DBX100.2	Oscillation reversal active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The deceleration period after external oscillation reversal (DB31, ...DBX28.0) is active
Signal state 0 or edge change 1 → 0	No deceleration after external oscillation reversal is active

DB31, ... DBX100.3	Oscillation cannot start
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The oscillation axis cannot be started owing to incorrect programming. This status can occur even when axis has already been traversed.
Signal state 0 or edge change 1 → 0	The oscillation movement can be started.

DB31, ... DBX100.4	Error during oscillation movement
Edge evaluation:	Signal(s) updated:
Signal state 1 or edge change 0 → 1	The oscillation movement has been aborted.
Signal state 0 or edge change 1 → 0	The oscillation movement is being executed correctly.

DB31, ... DBX100.5	Sparking-out active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is executing sparking-out strokes.
Signal state 0 or edge change 1 → 0	The axis is not currently executing sparking-out strokes.
Corresponding to	DBX100.7

DB31, ... DBX100.6	Oscillation movement active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is executing an oscillation movement between 2 reversal points.
Signal state 0 or edge change 1 → 0	The axis is not currently oscillating.
Signal irrelevant for	DBX100.7 = 0
Corresponding to	DBX100.7

DB31, ... DBX100.7	Oscillation active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis is currently being traversed as an oscillation axis.
Signal state 0 or edge change 1 → 0	The axis is a positioning axis.
Corresponding to	DBX100.5, DBX100.6

DB31, ... DBX104.0 - 7	Active infeed axes
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The axis sending the signal is currently the oscillation axis and is indicating its active infeed axes in this field (104.0 axis 1 is infeed axis, 104.1 axis 2 is infeed axis, etc.).
Signal state 0 or edge change 1 → 0	The associated axis is not an infeed axis.
Corresponding to	DBX100.7

19.13 Rotary axes

19.13.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX12.4	Traversing range limitation for modulo rotary axes
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Activate traversing range limitation for modulo rotary axes (software end switches, work field limitations).
Signal state 0 or edge change 1 → 0	Deactivate traversing range limitation for modulo rotary axes.
Signal irrelevant for ...	Linear axes/rotary axes without modulo functionality.
Application example(s)	Built-on rotary axis with monitoring

19.13.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX74.4	Monitoring status with modulo rotary axes
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	Traversing range limitation for modulo rotary axes active (software end switches, work field limitations).
Signal state 0 or edge change 1 → 0	Traversing range limitation for modulo rotary axes not active.
Signal irrelevant for ...	Linear axes/rotary axes without modulo functionality.
Application example(s)	Built-on rotary axis with monitoring

19.14 Synchronous spindle

19.14.1 Signals to axis/spindle (DB31, ...)

DB31, ... DBX31.5	Disable synchronization
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The synchronization motion for the following spindle is not disabled from the PLC. The position offset is not suppressed and applied as in earlier versions.
Signal state 0 or edge change 1 → 0	The synchronization motion for the following spindle is disabled from the PLC. A synchronization motion specified via offset programming is suppressed for the following spindle. The following spindle does not execute any additional movement.
Corresponding to	DB31, ... DBX98.1 (Synchronism coarse) DB31, ... DBX98.0 (Synchronism fine)

19.14.2 Signals from axis/spindle (DB31, ...)

DB31, ... DBX84.4	Synchronous mode
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The spindle is operating in "Synchronous operation" mode. The following spindle thus follows the movements of the leading spindle in accordance with the transmission ratio. The monitoring functions for coarse and fine synchronism are implemented in synchronous operation. Note: The signal is set only for the machine axis which is acting as following spindle (IS "FS active" = 1)
Signal state 0 or edge change 1 → 0	The spindle is not operated as the following spindle in "synchronous mode". When the coupling is deactivated (deselection of synchronous operation), the following spindle is switched to "open-loop control mode".
Corresponding to	DB31, ... DBX98.0 (Synchronism fine) DB31, ... DBX98.1 (Synchronism coarse) DB31, ... DBX99.1 (FS active)

DB31, ... DBX98.0	**Fine synchronism**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The positional deviation or velocity difference between the following spindle and its leading spindle is within the "Fine synchronism" tolerance band.
Signal state 0 or edge change 1 → 0	The positional deviation or velocity difference between the following spindle and its leading spindle is not within the "Fine synchronism" tolerance band. Note: The signal is relevant only for the following spindle in synchronous operation.
Application example	Clamping of workpiece in following spindle on transfer from the leading spindle: Clamping of the workpiece is not initiated by the PLC user program until the spindles are sufficiently synchronized.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) MD37210 $MA_COUPLE_POS_TOL_FINE (threshold value for "fine synchronism") MD37230 $MA_COUPLE_VELO_TOL_FINE ("fine" speed tolerance)

DB31, ... DBX98.1	**Coarse synchronism**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The positional deviation or velocity difference between the following spindle and its leading spindle is within the "Coarse synchronism" tolerance band. Note: The signal is relevant only for the following spindle in synchronous operation.
Signal state 0 or edge change 1 → 0	The positional deviation or velocity difference between the following spindle and its leading spindle is not within the "Coarse synchronism" tolerance band.
Application example	Clamping of workpiece in following spindle on transfer from the leading spindle: Clamping of the workpiece is not initiated by the PLC user program until the spindles are sufficiently synchronized.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) MD37200 $MA_COUPLE_POS_TOL_COARSE (threshold value for "coarse synchronism") MD37220 $MA_COUPLE_VELO_TOL_COARSE ("coarse" speed tolerance)

DB31, ... DBX98.2	**Actual value coupling**
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The actual-value coupling is active as the coupling type between the leading and following spindles (see MD21310). Note: The signal is relevant only for the active following spindle in synchronous operation.
Signal state 0 or edge change 1 → 0	The setpoint coupling is active as the coupling type between the leading and following spindles (see MD21310).
Special cases, errors, ...	In the case of faults/disturbances on the following spindle which result in cancellation of the FS "servo enable", the coupling relationship between the FS and LS is reversed and switched over to an actual-value coupling internally in the control under certain circumstances.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) MD21310 $MC_COUPLING_MODE_1 (coupling type in synchr. spindle oper.)

DB31, ... DBX98.4	Overlaid motion
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The following spindle traverses an additional motional component which is overlaid on the motion from the coupling with the leading spindle. Examples of overlaid movement of FS: - Activation of synchronous operation with defined angular offset between FS and LS - Activation of synchronous operation with LS in rotation - Alteration of transmission ratio when synchronous operation is selected - Input of a new defined angular offset when synchronous operation is selected - Traversal of FS with plus or minus traversing keys or handwheel in JOG when synchronous operation is selected As soon as the FS executes an overlaid movement, IS "Fine synchronism" or IS "Coarse synchronism" (depending on threshold value) may be canceled immediately. Note: The signal is relevant only for the following spindle in synchronous operation.
Signal state 0 or edge change 1 → 0	The following spindle does not traverse any additional motional component or this motion has been terminated.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode)

DB31, ... DBX99.0	LS (leading spindle) active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The machine axis is currently active as the leading spindle. Note: The signal is relevant only in synchronous operation.
Signal state 0 or edge change 1 → 0	The machine axis is not currently active as the leading spindle.
Special cases, errors, ...	In the case of faults/disturbances on the following spindle which result in cancellation of the FS "servo enable", the coupling relationship between the FS and LS is reversed and switched over to an actual-value coupling internally in the control under certain circumstances. In this case, the leading spindle becomes the new, active following spindle (IS "FS active").
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) DB31, ... DBX99.1 (FS active)

DB31, ... DBX99.1	FS (following spindle) active
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The machine axis is currently operating as the following spindle. The following spindle thus follows the movements of the leading spindle in synchronous operation in accordance with the transmission ratio. Note: The signal is relevant only in synchronous operation.
Signal state 0 or edge change 1 → 0	The machine axis is not currently operating as the following spindle.

DB31, ... DBX99.1	FS (following spindle) active
Special cases, errors, ...	In the case of faults/disturbances on the following spindle which result in cancellation of the FS "servo enable", the coupling relationship between the FS and LS is reversed and switched over to an actual-value coupling internally in the control under certain circumstances.
Corresponding to	DB31, ... DBX84.4 (Synchronous mode) DB31, ... DBX99.0 (LS active)

19.15 Memory configuration (S7)

No signal descriptions required.

19.16 Indexing axes

19.16.1 Signals from axis/spindle (DB31, ...)

DB31, ... DBX76.6	Indexing axis in position
Edge evaluation: No	Signal(s) updated: Cyclic
Signal state 1 or edge change 0 → 1	The signal is influenced according to the "Exact stop fine": When "Exact stop fine" is achieved, the signal is set. When exiting "Exact stop fine", the signal is reset. • The indexing axis is located on an indexing position. The indexing axis has been positioned with instructions for "Coded Position". **Note:** If the "Exact stop fine" window is reached and the indexing axis is positioned on an indexing position, the signal is enabled regardless of how the indexing position was reached.
Signal state 0 or edge change 1 → 0	• The axis is not defined as an indexing axis. • The indexing axis is traversing: DB31, ... DBX64.7/64.6 (Travel command+/-) is active. • The indexing axis is located at a position which is not an indexing position. Examples: – In JOG mode after abortion of travel movement, e.g. with RESET – in Automatic mode: indexing axis has, for example, approached a selected position controlled by an AC or DC instruction • The indexing axis has not been positioned with instructions for coded positions (CAC, CACP, CACN, CDC, CIC) in automatic mode. • The "Servo enable" signal for the indexing axis has been canceled: DB31, ... DBX2.1 (Servo enable)
Signal irrelevant for ...	... Axes that are not defined as indexing axes: MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB = 0

DB31, ... DBX76.6	Indexing axis in position
Application example(s)	Tool magazine: Activation of a gripper for removing a tool from a magazine is triggered when the indexing axis is in position: DB31, ... DBX76.6 (indexing axis in position) = 1. This must ensured by the PLC user program.
Special cases, errors, ...	**Notes:** The axis positions entered in the indexing position table for the individual divisions can be changed through zero offsets (including DRF). The interface signal: DB31, ... DBX76.6 (indexing axis in position) is then set to 1 when the actual position of the indexing axis matches the value entered in the index table plus the offset. If a DRF is applied to an indexing axis in AUTOMATIC mode, then interface signal "Indexing axis in position" remains active even though the axis is no longer at an indexing position.
Corresponding to	MD30500 $MA_INDEX_AX_ASSIGN_POS_TAB (axis is an indexing axis)

19.17 Tool change (W3)

No signal descriptions required.

19.18 Grinding-specific tool offset and tool monitoring

19.18.1 Signals from axis/spindle (DB31, ...)

DB31, ... DBX83.3	Geometry monitoring	
Edge evaluation: No		Signal(s) updated: -
Signal state 1 or edge change 0 → 1	Error in grinding wheel geometry. **Note:** There is no further reaction to the response of this monitoring function. Reactions deemed necessary must be programmed by the PLC user.	
Signal state 0 or edge change 1 → 0	No error in grinding wheel geometry.	
Application example(s)	Grinding-specific tool monitoring	

DB31, ... DBX83.6	Speed monitoring	
Edge evaluation: No		Signal(s) updated: -
Signal state 1 or edge change 0 → 1	Error in grinding wheel speed. **Note:** No further reaction to this signal state is programmed. Reactions deemed necessary must be programmed by the PLC user.	
Signal state 0 or edge change 1 → 0	No error in grinding wheel speed.	
Application example(s)	Grinding-specific tool monitoring	

DB31, ... DBX84.1	GWPS active	
Edge evaluation: No		Signal(s) updated: -
Signal state 1 or edge change 0 → 1	Constant grinding wheel peripheral speed (GWPS) is active. If GWPS is active, then all S value inputs from the PLC are interpreted as the grinding wheel peripheral speed.	
Signal state 0 or edge change 1 → 0	Constant grinding wheel peripheral speed (GWPS) is not active.	
Application example(s)	GWPS in all operating modes.	

Appendix

A

A.1 List of abbreviations

A	
A	Output
AC	Adaptive Control
ADI4	Analog Drive Interface for 4 Axes
ALM	Active Line Module
ARM	Rotating induction motor
AS	PLC
ASCII	American Standard Code for Information Interchange
ASUB	Asynchronous subroutine
AUXFU	Auxiliary Function

B	
BA	Mode
BAG	Mode group
BCD	Binary Coded Decimals
BCS	Basic Coordinate System
BERO	Proximity limit switch
BI	Binector Input
BICO	Binector Connector
BIN	BINary files
BO	Binector Output
BOT	Boot file (SIMODRIVE 611D)

C	
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CC	Compile Cycle
CF Card	Compact Flash Card
CI	Connector Input
CNC	Computerized Numerical Control
CO	Connector Output
CoL	Certificate of License
COM	Communication

CP	Communication Processor
CPA	Compiler Projecting Data: Configuring data of the compiler
CPU	Central Processing Unit
CR	Carriage Return
CRC	Cutter radius compensation
CTS	Clear To Send: Ready to send signal for serial data interfaces
CU	Control Unit
CUTCOM	Cutter radius Compensation: Tool radius compensation

D	
DAC	Digital-to-Analog Converter
DB	Data Block (PLC)
DBB	Data Block Byte (PLC)
DBD	Data Block Double word (PLC)
DBW	Data Block Word (PLC)
DBX	Data block bit (PLC)
DIN	Deutsche Industrie Norm (German Industry Standard)
DIO	Data Input/Output: Data transfer display
DIR	Directory
DO	Drive Object
DPM	Dual Port Memory
DPR	Dual Port RAM
DRAM	Dynamic memory (non-buffered)
DRF	Differential Resolver Function (handwheel)
DRIVE-CliQ	Drive Component Link with IQ
DRY	Dry Run: Dry run feedrate
DSB	Decoding Single Block
DSC	Dynamic Servo Control/Dynamic Stiffness Control
DW	Data Word
DWORD	Double Word (currently 32 bits)

E	
E	Input
EFP	Compact I/O module (PLC I/O module)
EMC	ElectroMagnetic Compatibility
EN	European standard
ENC	Encoder: Actual value encoder
EnDat	Encoder interface
EPROM	Erasable Programmable Read Only Memory
EQN	Designation for an absolute encoder with 2048 sine signals per revolution
ES	Engineering System

ESD	Electrostatic Sensitive Devices
ESR	Extended Stop and Retract
ETC	ETC key ">"; Softkey bar extension in the same menu

F	
FB	Function Block (PLC)
FC	Function Call: Function block (PLC)
FDD	Feed Drive
FEPROM	Flash EPROM: Read and write memory
FIFO	First In First Out: Memory that works without address specification and whose data are read in the same order in which they were stored.
FIPO	Fine interpolator
FRAME	Coordinate transformation
FST	Feed Stop
FW	Firmware

G	
GC	Global Control (PROFIBUS: Broadcast telegram)
GEO	Geometry, e.g. geometry axis
GIA	Gear Interpolation dAta
GND	Signal Ground
GP	Basic Program (PLC)
GS	Gear stage
GSD	Device master file for desribing a PROFIBUS slave
GSDML	Generic Station Description Markup Language: XML-based description language for creating a GSD file
GUD	Global User Data
GWPS	Grinding Wheel Peripheral Speed

H	
HEX	Abbreviation for hexadecimal number
HHU	HandHeld Unit
HMI	Human Machine Interface, SINUMERIK user interface
HW	Hardware
HW Config	SIMATIC S7 tool for configuration and parameterization of hardware components within an S7 project
HW limit switch	Hardware limit switch

I	
I/RF	Infeed/regenerative feedback unit of SIMODRIVE 611(D)
IBN	Commissioning
ICA	Interpolatory compensation
INC	Increment
IPO	Interpolator
IS	Interface Signal

J	
JOG	Jogging: Setup mode

K	
K_p	Proportional gain
$K_Ü$	Transformation ratio
K_V	Gain factor of control loop

L	
LAD	LADder diagram
LAN	Local Area Network
LEC	Leadscrew Error Compensation
LEDs	Light Emitting Diode
LF	Line Feed
LSB	Least Significant Bit
LUD	Local User Data

M	
MAC	Media Access Control
MB	Megabyte
MCI	Motion Control Interface
MCIS	Motion Control Information System
MCP	Machine Control Panel
MCS	Machine Coordinate System
MD	Machine Data
MDI	Manual Data Automatic: Manual input
MLFB	Machine-Readable Product Code
MM	Motor Module
MMC	Man Machine Communication
MPF	Main Program File: NC part program (main program)

MPI	Multi Port Interface
MSD	Main Spindle Drive
MSGW	Message word

N	
NC	Numerical Control
NCK	Numerical Control Kernel
NCU	Numerical Control Unit
NRK	Name for the operating system of the NCK
NST	Interface Signal
NX	Numerical eXtension (axis extension module)

O	
OB	Organization block in the PLC
OEM	Original Equipment Manufacturer
OLP	Optical Link Plug: Fiber optic bus connector
OP	Operator Panel: Operating equipment
OPI	Operator Panel Interface: Interface for connection to the operator panel
OPT	Options

P	
PC	Position Controller
PC	Personal Computer
PCMCIA	Personal Computer Memory Card International Association
PCU	PC Unit
PG	Programming device
PII	Process Image Input
PIQ	Process Image Output
PKE	Parameter identification: Part of a PIV
PKW	Parameter identification: Value: Parameterizing part of a PPO
PLC	Programmable Logic Control: Adaptation control
PMS	Position Measuring System
PN	PROFINET
PNO	PROFIBUS user organization
PO	POWER ON
POS	Positioning: e.g. POS axis = positioning axis = channel axis which is not traversed to its programmed position in an interpolatory relationship to other axes, i.e. it is traversed independently of other channel axes.
POSMO A	Positioning Motor Actuator: Positioning motor
POSMO CA	Positioning Motor Compact AC: Complete drive unit with integrated power and control module as well as positioning unit and program memory; AC infeed

POSMO CD	Positioning Motor Compact DC: Like CA but with DC infeed
POSMO SI	Positioning Motor Servo Integrated: Positioning motor, DC infeed
POU	Program Organization Unit
PPO	Parameter Process data Object; Cyclic data telegram for PROFIBUS DP transmission and "Variable speed drives" profile
PROFIBUS	Process Field Bus: Serial data bus
PRT	Program test
PSW	Program control word
PTP	Point-To-Point
PUD	Program global User Data: Global program variable
PZD	Process Data: Process data part of a PPO

Q	
QEC	Quadrant error compensation

R	
RAM	Random Access Memory: Read/write memory
REF	REFerence point approach function
REPOS	REPOSition function
RISC	Reduced Instruction Set Computer: Type of processor with small instruction set and ability to process instructions at high speed
ROV	Rapid Override: Input correction
RP	R variable, arithmetic parameter, predefined user variable
RPY	Roll Pitch Yaw: Rotation type of a coordinate system
RTCP	Real Time Control Protocol
RTLI	Rapid Traverse Linear Interpolation: Linear interpolation during rapid traverse motion

S	
SBC	Safe Brake Control
SBL	Single Block
SD	Setting Data
SEA	Setting Data Active: Identifier (file type) for setting data
SERUPRO	SEarch RUn by PROgram test
SGA	Safety-related output
SGE	Safety-related input
SH	Safe standstill
SIM	Single Inline Module
SK	Softkey
SKP	SKiP: Function for skipping a part program block
SLM	Synchronous Linear Motor
SM	Stepper motor

SMC	Sensor Module Cabinet Mounted
SME	Sensor Module Externally Mounted
SPF	Sub Program File
SRAM	Static RAM (non-volatile)
SRM	Synchronous Rotary Motor
SSI	Synchronous Serial Interface (interface type)
SSL	Block search
STW	Control word
SW	Software
SW limit switch	Software limit switch
SYF	System Files
SYNACT	SYNchronized ACTion

T	
T	Tool
TB	Terminal Board (SINAMICS)
TC	Tool Change
TCP	Tool Center Point: Tool tip
TCP/IP	Transport Control Protocol/Internet Protocol
TCU	Thin Client Unit
TEA	Testing Data Active: Identifier for machine data
TIA	Totally Integrated Automation
TLC	Tool Length Compensation
TM	Terminal Module (SINAMICS)
TM	Tool Management
TNRC	Tool Nose Radius Compensation
TO	Tool Offset
TOA	Tool Offset Active: Identifier (file type) for tool offsets
TRANSMIT	Transform Milling Into Turning: Coordination transformation for milling operations on a lathe
TRC	Tool Radius Compensation
TTL	Transistor-Transistor Logic (interface type)

U	
UP	User program
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus

V	
VDE	Verband Deutscher Elektrotechniker [Association of German Electrical Engineers]
VDI	Internal communication interface between NCK and PLC
VDI	Verein Deutscher Ingenieure [Association of German Engineers]
VI	Voltage Input
VO	Voltage Output

W	
WCS	Workpiece coordinate system
WO	Work Offset
WOA	Work Offset Active: Identifier for work offsets
WOP	Workshop-Oriented Programming
WPD	WorkPiece Directory

X	
XML	Extensible Markup Language

Z	
ZSW	Status word (of drive)

A.2 Feedback on the documentation

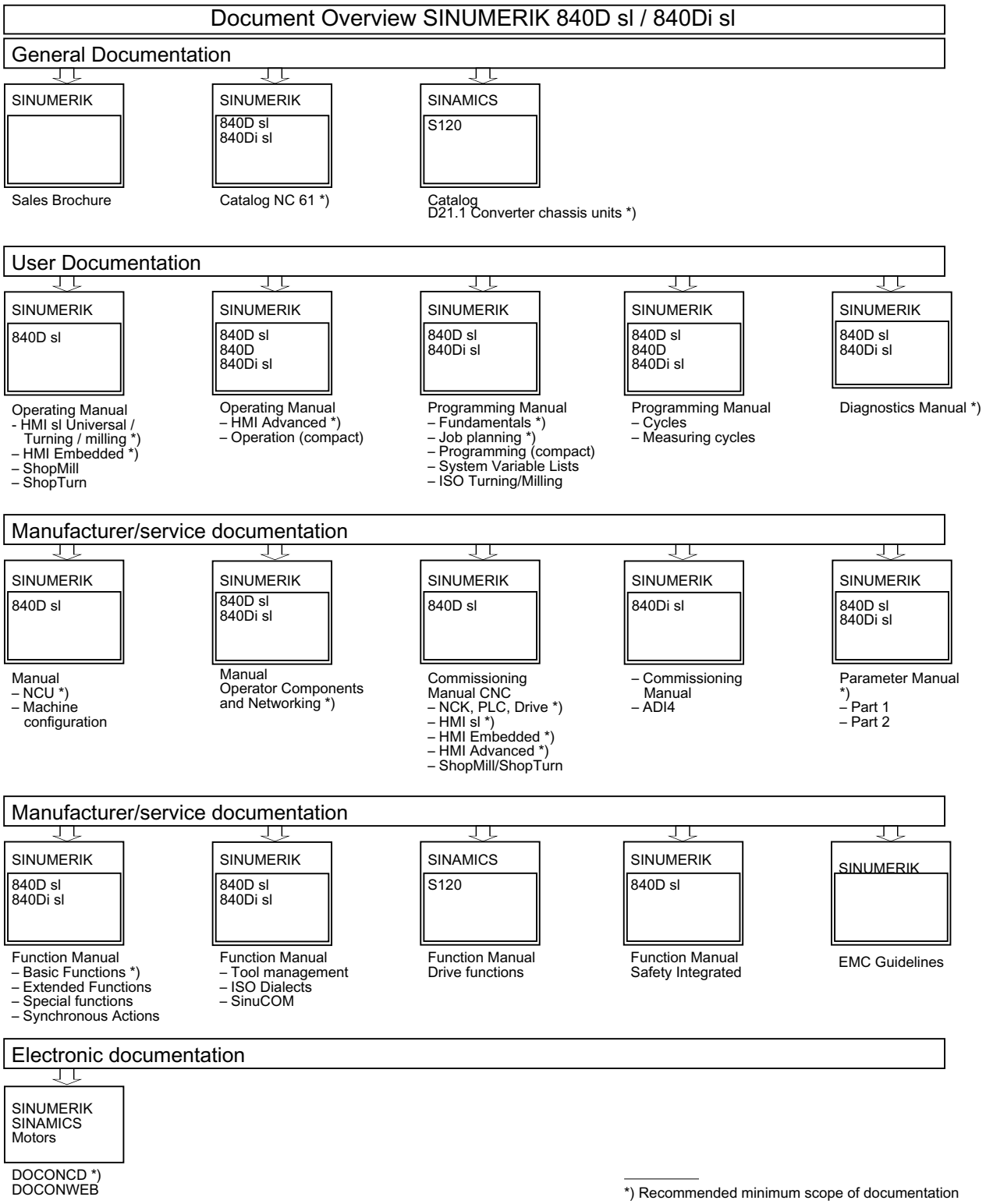
This document will be continuously improved with regard to its quality and ease of use. Please help us with this task by sending your comments and suggestions for improvement via e-mail or fax to:

E-mail:	mailto:docu.motioncontrol@siemens.com
Fax:	+49 9131 - 98 2176 Please use the fax form on the back of this page.

To	From
SIEMENS AG I DT MC MS1 P.O. Box 3180 D-91050 Erlangen / Germany	Name:
	Address of your company/department
	Street:
	Zip code: City:
	Phone: /
Fax: +49 9131 - 98 2176 (Documentation)	Fax: /

Suggestions and/or corrections

A.3 Overview


Document Overview SINUMERIK 840D sl / 840Di sl
General Documentation
SINUMERIK
Sales Brochure
SINUMERIK
840D sl
840Di sl
Catalog NC 61 *)
SINAMICS
S120
Catalog
D21.1 Converter chassis units *)
User Documentation
SINUMERIK
840D sl
Operating Manual
- HMI sl Universal / Turning / milling *)
– HMI Embedded *)
– ShopMill
– ShopTurn
SINUMERIK
840D sl
840D
840Di sl
Operating Manual
– HMI Advanced *)
– Operation (compact)
SINUMERIK
840D sl
840Di sl
Programming Manual
– Fundamentals *)
– Job planning *)
– Programming (compact)
– System Variable Lists
– ISO Turning/Milling
SINUMERIK
840D sl
840D
840Di sl
Programming Manual
– Cycles
– Measuring cycles
SINUMERIK
840D sl
840Di sl
Diagnostics Manual *)
Manufacturer/service documentation
SINUMERIK
840D sl
Manual
– NCU *)
– Machine configuration
SINUMERIK
840D sl
840Di sl
Manual
Operator Components and Networking *)
SINUMERIK
840D sl
Commissioning Manual CNC
– NCK, PLC, Drive *)
– HMI sl *)
– HMI Embedded *)
– HMI Advanced *)
– ShopMill/ShopTurn
SINUMERIK
840Di sl
– Commissioning Manual
– ADI4
SINUMERIK
840D sl
840Di sl
Parameter Manual *)
– Part 1
– Part 2
Manufacturer/service documentation
SINUMERIK
840D sl
840Di sl
Function Manual
– Basic Functions *)
– Extended Functions
– Special functions
– Synchronous Actions
SINUMERIK
840D sl
840Di sl
Function Manual
– Tool management
– ISO Dialects
– SinuCOM
SINAMICS
S120
Function Manual
Drive functions
SINUMERIK
840D sl
Function Manual
Safety Integrated
SINUMERIK
EMC Guidelines
Electronic documentation
SINUMERIK
SINAMICS
Motors
DOCONCD *)
DOCONWEB
*) Recommended minimum scope of documentation

Glossary

Absolute dimensions

A destination for an axis movement is defined by a dimension that refers to the origin of the currently active coordinate system. See → Incremental dimension

Acceleration with jerk limitation

In order to optimize the acceleration response of the machine whilst simultaneously protecting the mechanical components, it is possible to switch over in the machining program between abrupt acceleration and continuous (jerk-free) acceleration.

Address

An address is the identifier for a certain operand or operand range, e.g. input, output etc.

Alarms

All → messages and alarms are displayed on the operator panel in plain text with date and time and the corresponding symbol for the cancel criterion. Alarms and messages are displayed separately.

1. Alarms and messages in the part program:

   Alarms and messages can be displayed in plain text directly from the part program.

2. Alarms and messages from PLC

   Alarms and messages for the machine can be displayed in plain text from the PLC program. No additional function block packages are required for this purpose.

Archive

Reading out of files and/or directories on an **external** memory device.

Asynchronous subroutine

Part program that can be started asynchronously to (independently of) the current program status using an interrupt signal (e.g. "Rapid NC input" signal).

Automatic

Operating mode of the control (block sequence operation according to DIN): Operating mode for NC systems in which a → subprogram is selected and executed continuously.

Auxiliary functions

Auxiliary functions enable → part programs to transfer → parameters to the → PLC, which then trigger reactions defined by the machine manufacturer.

Axes

In accordance with their functional scope, the CNC axes are subdivided into:

- Axes: interpolating path axes
- Auxiliary axes: non-interpolating feed and positioning axes with an axis-specific feed rate. Auxiliary axes are not involved in actual machining, e.g. tool feeder, tool magazine.

Axis address

See → Axis identifier

Axis identifier

Axes are identifed using X, Y, and Z as defined in DIN 66217 for a dextrorotatory, right-angled → coordinate system.

Rotary axes rotating around X, Y, and Z are identified using A, B, and C. Additional axes situated parallel to the specified axes can be designated using other letters.

Axis name

See → Axis identifier

Backlash compensation

Compensation for a mechanical machine backlash, e.g. backlash on reversal for ball screws. Backlash compensation can be entered separately for each axis.

Backup battery

The backup battery ensures that the → user program in the → CPU is stored so that it is safe from power failure and so that specified data areas and bit memory, timers and counters are stored retentively.

Base axis

Axis whose setpoint or actual value position forms the basis of the calculation of a compensation value.

Basic coordinate system

Cartesian coordinate system which is mapped by transformation onto the machine coordinate system.

The programmer uses axis names of the basic coordinate system in the → part program. The basic coordinate system exists parallel to the → machine coordinate system if no → transformation is active. The difference between the two coordinate systems lies in the → axis identifiers.

Baud rate

Rate of data transfer (Bit/s).

Blank

Workpiece as it is before it is machined.

Block

"Block" is the term given to any files required for creating and processing programs.

Block search

For debugging purposes or following a program abort, the "Block search" function can be used to select any location in the part program at which the program is to be started or resumed.

Booting

Loading the system program after power ON.

C axis

Axis around which the tool spindle describes a controlled rotational and positioning movement.

Channel

A channel is characterized by the fact that it can process a → part program independently of other channels. A channel exclusively controls the axes and spindles assigned to it. Part program runs of different channels can be coordinated through → synchronization.

Circular interpolation

The → tool moves on a circle between specified points on the contour at a given feed rate, and the workpiece is thereby machined.

CNC

See → NC

COM

Component of the NC for the implementation and coordination of communication.

Compensation axis

Axis with a setpoint or actual value modified by the compensation value

Compensation memory

Data range in the control, in which the tool offset data are stored.

Compensation table

Table containing interpolation points. It provides the compensation values of the compensation axis for selected positions on the basic axis.

Compensation value

Difference between the axis position measured by the encoder and the desired, programmed axis position.

Connecting cables

Connecting cables are pre-assembled or user-assembled 2-wire cables with a connector at each end. This connecting cable connects the → CPU to a → programming device or to other CPUs by means of a → multi-point interface (MPI).

Continuous-path mode

The objective of continuous-path mode is to avoid substantial deceleration of the → path axes at the part program block boundaries and to change to the next block at as close to the same path velocity as possible.

Contour

Contour of the → workpiece

Contour monitoring

The following error is monitored within a definable tolerance band as a measure of contour accuracy. An unacceptably high following error can cause the drive to become overloaded, for example. In such cases, an alarm is output and the axes are stopped.

Coordinate system

See → Machine coordinate system, → Workpiece coordinate system

CPU

Central processing unit, see → PLC

C-Spline

The C-Spline is the most well-known and widely used spline. The transitions at the interpolation points are continuous, both tangentially and in terms of curvature. 3rd order polynomials are used.

Curvature

The curvature k of a contour is the inverse of radius r of the nestling circle in a contour point (k = 1/r).

Cycles

Protected subroutines for execution of repetitive machining operations on the → workpiece.

Data block

1. Data unit of the → PLC that → HIGHSTEP programs can access.
2. Data unit of the → NC: Data modules contain data definitions for global user data. These data can be initialized directly when they are defined.

Data word

Two-byte data unit within a → data block.

Diagnosis

1. Operating area of the control.
2. The control has both a self-diagnostics program as well as test functions for servicing purposes: status, alarm, and service displays

Dimensions specification, metric and inches

Position and lead values can be programmed in inches in the machining program. Irrespective of the programmable dimensions (G70/G71), the controller is set to a basic system.

DRF

Differential Resolver Function: NC function which generates an incremental zero offset in Automatic mode in conjunction with an electronic handwheel.

Drive

The drive is the unit of the CNC that performs the speed and torque control based on the settings of the NC.

Dynamic feedforward control

Inaccuracies in the → contour due to following errors can be practically eliminated using dynamic, acceleration-dependent feedforward control. This results in excellent machining accuracy even at high → path velocities. Feedforward control can be selected and deselected on an axis-specific basis via the → part program.

Editor

The editor makes it possible to create, edit, extend, join, and import programs/texts/program blocks.

Exact stop

When an exact stop statement is programmed, the position specified in a block is approached exactly and, if necessary, very slowly. To reduce the approach time, → exact stop limits are defined for rapid traverse and feed.

Exact stop limit

When all path axes reach their exact stop limits, the control responds as if it had reached its precise destination point. A block advance of the → part program occurs.

External zero offset

Zero offset specified by the → PLC.

Fast retraction from contour

When an interrupt occurs, a motion can be initiated via the CNC machining program, enabling the tool to be quickly retracted from the workpiece contour that is currently being machined. The retraction angle and the distance retracted can also be parameterized. After fast retraction, an interrupt routine can also be executed (SINUMERIK 840D).

Feed override

The programmed velocity is overriden by the current velocity setting made via the → machine control panel or from the → PLC (0 to 200%). The feedrate can also be corrected by a programmable percentage factor (1-200%) in the machining program.

Finished-part contour

Contour of the finished workpiece. See → Raw part.

Fixed machine point

Point that is uniquely defined by the machine tool, e.g. machine reference point.

Fixed-point approach

Machine tools can approach fixed points such as a tool change point, loading point, pallet change point, etc. in a defined way. The coordinates of these points are stored in the control. The control moves the relevant axes in → rapid traverse, whenever possible.

Frame

A frame is an arithmetic rule that transforms one Cartesian coordinate system into another Cartesian coordinate system. A frame contains the following components: → zero offset, → rotation, → scaling, → mirroring.

Geometry

Description of a → workpiece in the → workpiece coordinate system.

Geometry axis

Geometry axes are used to describe a 2- or 3-dimensional area in the workpiece coordinate system.

Ground

Ground is taken as the total of all linked inactive parts of a device which will not become live with a dangerous contact voltage even in the event of a malfunction.

Helical interpolation

The helical interpolation function is ideal for machining internal and external threads using form milling cutters and for milling lubrication grooves.

The helix comprises two movements:

- Circular movement in one plane
- A linear movement perpendicular to this plane

High-level CNC language

The high-level language offers: → user-defined variables, → system variables, → macro techniques.

High-speed digital inputs/outputs

The digital inputs can be used for example to start fast CNC program routines (interrupt routines). The digital CNC outputs can be used to trigger fast, program-controlled switching functions (SINUMERIK 840D).

HIGHSTEP

Summary of programming options for → PLCs of the AS300/AS400 system.

Identifier

In accordance with DIN 66025, words are supplemented using identifiers (names) for variables (arithmetic variables, system variables, user variables), subroutines, key words, and words with multiple address letters. These supplements have the same meaning as the words with respect to block format. Identifiers must be unique. It is not permissible to use the same identifier for different objects.

Inch measuring system

Measuring system, which defines distances in inches and fractions of inches.

Inclined surface machining

Drilling and milling operations on workpiece surfaces that do not lie in the coordinate planes of the machine can be performed easily using the function "inclined-surface machining".

Increment

Travel path length specification based on number of increments. The number of increments can be stored as → setting data or be selected by means of a suitably labeled key (i.e. 10, 100, 1000, 10000).

Incremental dimension

Also incremental dimension: A destination for axis traversal is defined by a distance to be covered and a direction referenced to a point already reached. See → Absolute dimension.

Intermediate blocks

Motions with selected → tool offset (G41/G42) may be interrupted by a limited number of intermediate blocks (blocks without axis motions in the offset plane), whereby the tool offset can still be correctly compensated for. The permissible number of intermediate blocks which the control reads ahead can be set in system parameters.

Interpolator

Logic unit of the → NCK that defines intermediate values for the motions to be carried out in individual axes based on information on the end positions specified in the part program.

Interpolatory compensation

Interpolatory compensation is a tool that enables manufacturing-related leadscrew error and measuring system error compensations (SSFK, MSFK).

Interrupt routine

Interrupt routines are special → subroutines that can be started by events (external signals) in the machining process. A part program block which is currently being worked through is interrupted and the position of the axes at the point of interruption is automatically saved.

Inverse-time feedrate

With SINUMERIK 840D, the time required for the path of a block to be traversed can be programmed for the axis motion instead of the feed velocity (G93).

JOG

Control operating mode (setup mode): In JOG mode, the machine can be set up. Individual axes and spindles can be traversed in JOG mode by means of the direction keys. Additional functions in JOG mode include: → Reference point approach, → Repos, and → Preset (set actual value).

Key switch

The key switch on the → machine control panel has four positions that are assigned functions by the operating system of the control. The key switch has three different colored keys that can be removed in the specified positions.

Keywords

Words with specified notation that have a defined meaning in the programming language for → part programs.

KV

Servo gain factor, a control variable in a control loop.

Leading axis

The leading axis is the → gantry axis that exists from the point of view of the operator and programmer and, thus, can be influenced like a standard NC axis.

Leadscrew error compensation

Compensation for the mechanical inaccuracies of a leadscrew participating in the feed. The control uses stored deviation values for the compensation.

Limit speed

Maximum/minimum (spindle) speed: The maximum speed of a spindle can be limited by specifying machine data, the → PLC or → setting data.

Linear axis

In contrast to a rotary axis, a linear axis describes a straight line.

Linear interpolation

The tool travels along a straight line to the destination point while machining the workpiece.

Load memory

The load memory is the same as → RAM for the CPU 314 of the → PLC.

Look Ahead

The **Look Ahead** function is used to achieve an optimal machining speed by looking ahead over an assignable number of traversing blocks.

Machine axes

Physically existent axes on the machine tool.

Machine control panel

An operator panel on a machine tool with operating elements such as keys, rotary switches, etc., and simple indicators such as LEDs. It is used to directly influence the machine tool via the PLC.

Machine coordinate system

A coordinate system, which is related to the axes of the machine tool.

Machine zero

Fixed point of the machine tool to which all (derived) measuring systems can be traced back.

Machining channel

A channel structure can be used to shorten idle times by means of parallel motion sequences, e.g. moving a loading gantry simultaneously with machining. Here, a CNC channel must be regarded as a separate CNC control system with decoding, block preparation and interpolation.

Macro techniques

Grouping of a set of statements under a single identifier. The identifier represents the set of consolidated statements in the program.

Main block

A block prefixed by ":" introductory block, containing all the parameters required to start execution of a -> part program.

Main program

The → part program designated by a number or an identifer in which additional main programs, subroutines, or → cycles can be called.

MDA

Control operating mode: Manual Data Automatic. In the MDA mode, individual program blocks or block sequences with no reference to a main program or subroutine can be input and executed immediately afterwards through actuation of the NC start key.

Messages

All messages programmed in the part program and → alarms detected by the system are displayed on the operator panel in plain text with date and time and the corresponding symbol for the cancel criterion. Alarms and messages are displayed separately.

Metric measuring system

Standardized system of units: For length, e.g. mm (millimeters), m (meters).

Mirroring

Mirroring reverses the signs of the coordinate values of a contour, with respect to an axis. It is possible to mirror with respect to more than one axis at a time.

Mode group

Axes and spindles that are technologically related can be combined into one mode group. Axes/spindles of a BAG can be controlled by one or more → channels. The same → mode type is always assigned to the channels of the mode group.

Mode of operation

An operating concept on a SINUMERIK control. The following modes are defined: → Jog, → MDA, → Automatic.

NC

Numerical Control: Numerical control (NC) includes all components of machine tool control: → NCK, → PLC, HMI, → COM.

Note

A more correct term for SINUMERIK 840D controls would be: Computerized Numerical Control

NCK

Numerical Control Kernel: Component of NC that executes the → part programs and basically coordinates the motion operations for the machine tool.

Network

A network is the connection of multiple S7-300 and other end devices, e.g. a programming device via a → connecting cable. A data exchange takes place over the network between the connected devices.

NRK

Numeric robotic kernel (operating system of → NCK)

NURBS

The motion control and path interpolation that occurs within the control is performed based on NURBS (**N**on **U**niform **R**ational **B**-**S**plines). As a result, a uniform process is available within the control for all interpolations for SINUMERIK 840D.

OEM

The scope for implementing individual solutions (OEM applications) for the SINUMERIK 840D has been provided for machine manufacturers, who wish to create their own operator interface or integrate process-oriented functions in the control.

Operator Interface

The user interface (UI) is the display medium for a CNC in the form of a screen. It features horizontal and vertical softkeys.

Oriented spindle stop

Stops the workpiece spindle in a specified angular position, e.g. in order to perform additional machining at a particular location.

Oriented tool retraction

`RETTOOL`: If machining is interrupted (e.g. when a tool breaks), a program command can be used to retract the tool in a user-specified orientation by a defined distance.

Overall reset

In the event of an overall reset, the following memories of the → CPU are deleted:

- → Work memory
- Read/write area of → load memory
- → System memory
- → Backup memory

Override

Manual or programmable control feature, which enables the user to override programmed feedrates or speeds in order to adapt them to a specific workpiece or material.

Part program block

Part of a → part program that is demarcated by a line feed. There are two types: → main blocks and → subblocks.

Part program management

Part program management can be organized by → workpieces. The size of the user memory determines the number of programs and the amount of data that can be managed. Each file (programs and data) can be given a name consisting of a maximum of 24 alphanumeric characters.

Path axis

Path axes include all machining axes of the → channel that are controlled by the → interpolator in such a way that they start, accelerate, stop, and reach their end point simultaneously.

Path feedrate

Path feed affects → path axes. It represents the geometric sum of the feed rates of the → geometry axes involved.

Path velocity

The maximum programmable path velocity depends on the input resolution. For example, with a resolution of 0.1 mm the maximum programmable path velocity is 1000 m/min.

PCIN data transfer program

PCIN is an auxiliary program for sending and receiving CNC user data (e.g. part programs, tool offsets, etc.) via a serial interface. The PCIN program can run in MS-DOS on standard industrial PCs.

Peripheral module

I/O modules represent the link between the CPU and the process.

I/O modules are:

- → Digital input/output modules
- → Analog input/output modules
- → Simulator modules

PLC

Programmable **L**ogic **C**ontrol: → Programmable logic controller. Component of → NC: Programmable controller for processing the control logic of the machine tool.

PLC program memory

SINUMERIK 840D: The PLC user program, the user data and the basic PLC program are stored together in the PLC user memory.

PLC programming

The PLC is programmed using the **STEP 7** software. The STEP 7 programming software is based on the **WINDOWS** standard operating system and contains the STEP 5 programming functions with innovative enhancements.

Polar coordinates

A coordinate system, which defines the position of a point on a plane in terms of its distance from the origin and the angle formed by the radius vector with a defined axis.

Polynomial interpolation

Polynomial interpolation enables a wide variety of curve characteristics to be generated, such as **straight line, parabolic, exponential functions** (SINUMERIK 840D).

Positioning axis

Axis that performs an auxiliary movement on a machine tool (e.g. tool magazine, pallet transport). Positioning axes are axes that do not interpolate with → path axes.

Position-time cams

The term "position-time cam" refers to a pair of software cams that can supply a pulse of a certain duration at a defined axis position.

Pre-coincidence

Block change occurs already when the path distance approaches an amount equal to a specifiable delta of the end position.

Program block

Program blocks contain the main program and subroutines of → part programs.

Programmable frames

Programmable → frames enable dynamic definition of new coordinate system output points while the part program is being executed. A distinction is made between absolute definition using a new frame and additive definition with reference to an existing starting point.

Programmable Logic Control

Programmable logic controllers (PLC) are electronic controls, the function of which is stored as a program in the control unit. This means that the layout and wiring of the device do not depend on the function of the control. The programmable logic controller has the same structure as a computer; it consists of a CPU (central module) with memory, input/output modules and an internal bus system. The peripherals and the programming language are matched to the requirements of the control technology.

Programmable working area limitation

Limitation of the motion space of the tool to a space defined by programmed limitations.

Programming key

Character and character strings that have a defined meaning in the programming language for → part programs.

Protection zone

Three-dimensional zone within the → working area into which the tool tip must not pass.

Quadrant error compensation

Contour errors at quadrant transitions, which arise as a result of changing friction conditions on the guideways, can be virtually entirely eliminated with the quadrant error compensation. Parameterization of the quadrant error compensation is performed by means of a circuit test.

R parameters

Arithmetic parameter that can be set or queried by the programmer of the → part program for any purpose in the program.

Rapid traverse

The highest traverse rate of an axis. For example, rapid traverse is used when the tool approaches the → workpiece contour from a resting position or when the tool is retracted from the workpiece contour. The rapid traverse velocity is set on a machine-specific basis using a machine data element.

Reference point

Machine tool position that the measuring system of the → machine axes references.

Rotary axis

Rotary axes apply a workpiece or tool rotation to a defined angular position.

Rotation

Component of a → frame that defines a rotation of the coordinate system around a particular angle.

Rounding axis

Rounding axes rotate a workpiece or tool to an angular position corresponding to an indexing grid. When a grid index is reached, the rounding axis is "in position".

Safety functions

The control is equipped with permanently active montoring functions that detect faults in the → CNC, the → PLC, and the machine in a timely manner so that damage to the workpiece, tool, or machine is largely prevented. In the event of a fault, the machining operation is interrupted and the drives stopped. The cause of the malfunction is logged and output as an alarm. At the same time, the PLC is notified that a CNC alarm has been triggered.

Scaling

Component of a → frame that implements axis-specific scale modifications.

Selecting

Series of statements to the NC that act in concert to produce a particular → workpiece. Likewise, this term applies to execution of a particular machining operation on a given → raw part.

Serial RS-232-C interface

For data input/output, the PCU 20 has one serial V.24 interface (RS232) while the PCU 50/70 has two V.24 interfaces. Machining programs and manufacturer and user data can be loaded and saved via these interfaces.

Setting data

Data, which communicates the properties of the machine tool to the NC, as defined by the system software.

Softkey

A key, whose name appears on an area of the screen. The choice of soft keys displayed is dynamically adapted to the operating situation. The freely assignable function keys (soft keys) are assigned defined functions in the software.

Software limit switch

Software limit switches limit the traversing range of an axis and prevent an abrupt stop of the slide at the hardware limit switch. Two value pairs can be specified for each axis and activated separately by means of the → PLC.

Spline interpolation

With spline interpolation, the controller can generate a smooth curve characteristic from only a few specified interpolation points of a set contour.

SRT

Transformation ratio

Standard cycles

Standard cycles are provided for machining operations, which are frequently repeated:

- Cycles for drilling/milling applications
- for turning technology

The available cycles are listed in the "Cycle support" menu in the "Program" operating area. Once the desired machining cycle has been selected, the parameters required for assigning values are displayed in plain text.

Subblock

Block preceded by "N" containing information for a sequence, e.g. positional data.

Subroutine

Sequence of statements of a → part program that can be called repeatedly with different defining parameters. The subroutine is called from a main program. Every subroutine can be protected against unauthorized read-out and display. → Cycles are a form of subroutines.

Synchronization

Statements in → part programs for coordination of sequences in different → channels at certain machining points.

Synchronized actions

1. Auxiliary function output

   During workpiece machining, technological functions (→ auxiliary functions) can be output from the CNC program to the PLC. For example, these auxiliary functions are used to control additional equipment for the machine tool, such as quills, grabbers, clamping chucks, etc.

2. Fast auxiliary function output

   For time-critical switching functions, the acknowledgement times for the → auxiliary functions can be minimized and unnecessary hold points in the machining process can be avoided.

Synchronized axes

Synchronized axes take the same time to traverse their path as the geometry axes take for their path.

Synchronized axis

A synchronized axis is the → gantry axis whose set position is continuously derived from the motion of the → leading axis and is, thus, moved synchronously with the leading axis. From the point of view of the programmer and operator, the synchronized axis "does not exist".

System memory

The system memory is a memory in the CPU in which the following data is stored:

- Data required by the operating system
- The operands times, counters, markers

System variables

A variable that exists without any input from the programmer of a → part program. It is defined by a data type and the variable name preceded by the character **$**. See → User-defined variable.

Tapping without compensating chuck

This function allows threads to be tapped without a compensating chuck. By using the interpolating method of the spindle as a rotary axis and the drilling axis, threads can be cut to a precise final drilling depth, e.g. for blind hole threads (requirement: spindles in axis operation).

Text editor

See → Editor

TOA area

The TOA area includes all tool and magazine data. By default, this area coincides with the → channel area with regard to the reach of the data. However, machine data can be used to specify that multiple channels share one → TOA unit so that common tool management data is then available to these channels.

TOA unit

Each → TOA area can have more than one TOA unit. The number of possible TOA units is limited by the maximum number of active → channels. A TOA unit includes exactly one tool data block and one magazine data block. In addition, a TOA unit can also contain a toolholder data block (optional).

Tool

Active part on the machine tool that implements machining (e.g. turning tool, milling tool, drill, LASER beam, etc.).

Tool nose radius compensation

Contour programming assumes that the tool is pointed. Because this is not actually the case in practice, the curvature radius of the tool used must be communicated to the control which then takes it into account. The curvature center is maintained equidistantly around the contour, offset by the curvature radius.

Tool offset

Consideration of the tool dimensions in calculating the path.

Tool radius compensation

To directly program a desired → workpiece contour, the control must traverse an equistant path to the programmed contour taking into account the radius of the tool that is being used (G41/G42).

Transformation

Additive or absolute zero offset of an axis.

Traversing range

The maximum permissible travel range for linear axes is ± 9 decades. The absolute value depends on the selected input and position control resolution and the unit of measurement (inch or metric).

User memory

All programs and data, such as part programs, subroutines, comments, tool offsets, and zero offsets/frames, as well as channel and program user data, can be stored in the shared CNC user memory.

User program

User programs for the S7-300 automation systems are created using the programming language STEP 7. The user program has a modular layout and consists of individual blocks.

The basic block types are:

- Code blocks

  These blocks contain the STEP 7 commands.

- Data blocks

  These blocks contain constants and variables for the STEP 7 program.

User-defined variable

Users can declare their own variables for any purpose in the → part program or data block (global user data). A definition contains a data type specification and the variable name. See → System variable.

Variable definition

A variable definition includes the specification of a data type and a variable name. The variable names can be used to access the value of the variables.

Velocity control

In order to achieve an acceptable traverse rate in the case of very slight motions per block, an anticipatory evaluation over several blocks (→ Look Ahead) can be specified.

WinSCP

WinSCP is a freely available open source program for Windows for the transfer of files.

Working area

Three-dimensional zone into which the tool tip can be moved on account of the physical design of the machine tool. See → Protection zone.

Working area limitation

With the aid of the working area limitation, the traversing range of the axes can be further restricted in addition to the limit switches. One value pair per axis may be used to describe the protected working area.

Working memory

RAM is a work memory in the → CPU that the processor accesses when processing the application program.

Workpiece

Part to be made/machined by the machine tool.

Workpiece contour

Set contour of the → workpiece to be created or machined.

Workpiece coordinate system

The workpiece coordinate system has its starting point in the → workpiece zero-point. In machining operations programmed in the workpiece coordinate system, the dimensions and directions refer to this system.

Workpiece zero

The workpiece zero is the starting point for the → workpiece coordinate system. It is defined in terms of distances to the → machine zero.

Zero offset

Specifies a new reference point for a coordinate system through reference to an existing zero point and a → frame.

1. Settable

   SINUMERIK 840D: A configurable number of settable zero offsets are available for each CNC axis. The offsets - which are selected by means of G functions - take effect alternately.

2. External

   In addition to all the offsets which define the position of the workpiece zero, an external zero offset can be overridden by means of the handwheel (DRF offset) or from the PLC.

3. Programmable

   Zero offsets can be programmed for all path and positioning axes using the `TRANS` statement.

Index

$

$A_DPx_IN, 61
$A_DPx_OUT, 61
$A_OUT[n], 708
$AA_ACT_INDEX_AX_POS_NO, 949
$AA_COUP_ACT, 897
$AA_COUP_OFFS, 897
$AA_ENC_COMP, 361
$AA_ENC_COMP_IS_MODULO, 362
$AA_ENC_COMP_MAX, 362
$AA_ENC_COMP_MIN, 361
$AA_ENC_COMP_STEP, 361
$AA_FIX_POINT_ACT, 335
$AA_FIX_POINT_SELECTED, 335
$AA_G0MODE, 773
$AA_PROG_INDEX_AX_POS_NO, 949
$AC_AXCTSWA[ ], 156
$AN_AXCTAS[ ], 156
$AN_AXCTSWA[ ], 156
$AN_CEC, 369
$AN_CEC_DIRECTION, 370
$AN_CEC_INPUT_AXIS, 369
$AN_CEC_IS_MODULO, 371
$AN_CEC_MAX, 370
$AN_CEC_MIN, 369
$AN_CEC_MULT_BY_TABLE, 371
$AN_CEC_OUTPUT_AXIS, 369
$AN_CEC_STEP, 369
$AN_REBOOT_DELAY_TIME, 454
$P_COUP_OFFS, 898
$VA_COUP_OFFS, 897

1

1-dimensional
    Setpoint selection ($AC_MEAS_TYPE = 19), 655

2

2-dimensional
    Setpoint selection ($AC_MEAS_TYPE = 20), 657

3

3D Probe, 622
3-dimensional
    Setpoint selection ($AC_MEAS_TYPE = 21), 658

A

Acceleration, 286, 298, 328
Acceleration characteristic, 731
Acknowledgement of manual stroke initiation, 1071
Acknowledgement of stopped status, 1074
ACN, 858
ACP, 858
Activate DRF, 1040
Activate handwheel (1 to 3) for geometry axis (1, 2, 3), 1040
Activate handwheel 1 as contour handwheel, 1050
Activate handwheel 2 as contour handwheel, 1050
Activate handwheel 3 as contour handwheel, 1050
Activation, 561
Activation methods, 883
Activation of coupling, 882
Active file system, 926
Active infeed axes, 1079
Active/passive operating mode, 1032
Active/passive operating mode of control unit, 1032
Actual value coupling, 1081
Actual value for analog NCK inputs, 1023
Actual value for digital NCK inputs, 1023
Alarm
    Text file, 92
    Text management, 92
Alarms, 94
Alarms,
    Messages, 92
All transformations, 580
Alter reversal point, 1076
Alternate interface, 729
Ambiguity in position
    Examples, 566
Ambiguity in rotary axis position
    Example, 567
AMIRROR(C), 868
Analog I/Os|840Di, 29
Analog-value representation, 52

Angle
Inclined axis, 544
Angular offset POSFS, 883
Applications, 550, 941
Approaching a fixed point
in JOG, 331
With G75, 331
ASCALE, 865
Assign feedrate using the programmed axis name of a positioning axis, 1075
Assignment
Bus nodes - bus system, 88
by axis group, 101
HMI - PCU, 88
Assignment of channel axes, 505
Assignment of channel axes to machine axes, 529
Assignment of geometry axes to channel axes, 529, 543
Assignment of geometry axes to channel axes, 505
Assignment of names to, 506
Assignment of names to geometry axes, 504, 528
ATRANS, 865
Automatic axis replacement with GETD, 478
Automatically activated pre-initiation time, 730
Autonomous machine, 97
Autonomous single-axis operations
NCK reactions, 802
PLC actions, 801
AxAlarm, 1073
Axes
for auxiliary movements, 763
Axial reset has been performed, 1074
Axial stop alarm for this axis, 1073
Axis
Basic, 355
Compensation, 355
Holding the workpiece, 101
Interpolator, 771
Local, 101, 102
Physical, 101
Workpiece-holding, 102
Axis configuration, 525, 528, 540, 542, 550
Axis container, 101, 102, 155, 391
Axis container names, 156
Axis container rotation, 158
Axis container rotation active, 1035
Axis control passed to PLC, 1074
Axis image, 543
Axis ready, 1035
Axis replacement, 470
Automatically generated GET/GETD, 478
Axis in another channel, 471
Axis replacement via synchronized actions, 489
Axis types, 470
Geometry axis in rotated frame, 488
Release axis container rotation, 484
Requirements, 471
Time of release, 483
without preprocessing stop, 485
Axis replacement via PLC, 800
Axis types
For positioning axes, 768
Axis/spindle interchange, 460
Axis/spindle replacement, 470, 1062
AxReset, 1073
AXRESET DONE, 1074
AxResume, 1073
AxStop active, 1074
AXTOCHAN, 489

B

Backlash, 353
Compensation, 352
Mechanical, 352
Backup battery, 92
Bidirectional probe, 607
Block change
Positioning axis type 1, 788
Positioning axis type 2, 790
Positioning axis type 3, 791
Block change criterion
Braking ramp with tolerance window, 794
Deceleration ramp, 793
Block search, 733
Bus
Nodes, 85, 87, 90
System, 85, 86

C

Calculated frame, 620
Calculation method, 625
Cam activation, 1067
Cam positions
Axis/cam assignment, 704
Setting cam positions, 702
Cam range/cam pair, 693
Cam signal output
Independent, timer-controlled, 710
Cam signals
Activation of signal output, 706
for linear axes, 695, 699
for modulo rotary axes, 697
Hardware assignment, 707

Lead/delay times, 704
Minus, 706
Output to NCK I/Os, 708
Output to PLC, 706
Plus, 707
Cams active, 1067
Cartesian manual travel, 568
Cartesian PTP travel, 561
STAT address, 564
TU address, 565
CC-Bindings, 64
Chained transformations, 550, 593
Activating, 552
Amount, 551
Deactivation, 552
Example, 591
Persistent transformation, 553
Special points to be noted, 552
Chaining direction, 551
Chaining rule, 982
Changing the assignment, 579
Channel, 459, 766
Menu, 93
Name, 92
Channel axis, 470
Channel axis identifier, 390
Channel number geometry axis for handwheel 1, 2, 3, 1036
Channel synchronization, 462
Characteristics, 577
Circularity test, 410, 412, 446
Display, 448
Measurement, 447
Parameterization, 446
Representation, 447
Clamping axis/spindle, 102
Clamping protection zone, 753
CLEAR, 463
CLEARM, 463
COARSA, 792
Coded positions, 944
Combination of different bus systems, 88
Command axes, 792
Axis interpolator, 792
Commands MEAS, MEAW, 613
Commissioning of neural QEC, 434
Comparator inputs, 29, 54
Compensation
Angularity error, 364
Following error, 399
Interpolatory, 355
Leadscrew error, 359
Measuring system error, 359
Sag, 364
Complete machining, 502
Computing capacity, 98
Concurrent positioning axes
Start from the PLC, 801
Conditional waiting
Accelerate from standstill to path velocity, 466
In continuous-path mode, 465
in the breaking ramp, depending on IPO step, 466
No wait., 466
Configuration file, 114, 135
NETNAMES.INI, 221
conn_1, 91
Connection operating area, 95
Connection to 840D, 608
Connection to SINUMERIK 840D sl, 609
Connection to SINUMERIK 840Di, 609
Constraints, 555
Container axes, 155
Container link axes, 155
Continuous dressing, 991
Continuous move, 942
Continuous operation, 942
Continuous travel, 289
Continuous traversing
Continuous operation, 289
Jog mode, 289
Contour handwheel, 312
Contour-handwheel simulation ON, 1051
Contour-handwheel-simulation negative direction, 1051
Control
Unit, 86, 89
Control of manual-travel functions, 287
Control unit requests active operating mode, 1032
Control unit switchover disable, 1031
Conversion into another coordinate system, 621
Coordinate systems, 286
Corner C1 - C4 ($AC_MEAS_TYPE = 4, 5, 6, 7), 634
Corner measurement C1, 635
COROS OP, 87
Coupling
Define new, 893
Fixed configuration, 892
Coupling options, 877
Cut-to-cut time, 968
Cylinder coordinate system, 525
Cylinder generated surface, 498

D

Data
Backup via V.24, 91

Exchange, 90
Data access, 59
Read access, 59
Write access, 59
Data backup
via V.24, 95
Data consistency, 61
DB 31, ...
DBX2.0, 707
DBX62.0, 707
DB10
DBB0, 35, 1016
DBB1, 35, 36, 1016
DBB100, 1037
DBB100 ff, 296
DBB101, 1037
DBB102, 1037
DBB122, 35, 1016
DBB123, 35, 36, 1016
DBB124, 1016
DBB125, 1016
DBB126, 1016
DBB127, 1016
DBB128, 1016
DBB129, 1016
DBB130, 38, 1016
DBB131, 38, 1017
DBB132, 38, 1017
DBB133, 38, 1018
DBB134, 1016
DBB135, 1017
DBB136, 1017
DBB137, 1018
DBB138, 1016
DBB139, 1017
DBB140, 1017
DBB141, 1018
DBB142, 1016
DBB143, 1017
DBB144, 1017
DBB145, 1018
DBB146, 43, 1018
DBB147, 43, 1018
DBB148 - 163, 43
DBB148 -163, 1019
DBB166, 46, 1019
DBB167, 46, 1019
DBB168, 46, 1020
DBB170 - 185, 46
DBB170 -185, 1020
DBB186, 39
DBB186-189, 1023
DBB190-193, 1023
DBB194 - 209, 43
DBB194 -209, 1023
DBB210 - 225, 47
DBB210 -225, 1024
DBB4, 38, 1016
DBB5, 38, 1017
DBB6, 38, 1017
DBB60, 1023
DBB64, 39, 1023
DBB7, 38, 1018
DBB97, 1036
DBB98, 1036
DBB99, 1036
DBX100.6, 1038
DBX100.6 ff, 296
DBX100.7, 1038
DBX100.7 ff, 296
DBX101.6, 1038
DBX101.7, 1038
DBX102.6, 1038
DBX102.7, 1038
DBX107.0, 1064
DBX107.1, 1064
DBX107.6, 1034
DBX108.7, 453
DBX110.0 - 113.7, 706, 707
DBX110.0-113.7, 1066
DBX114.0 - 117.7, 707
DBX114.0-117.7, 1066
DBX97 ff, 296
DB11
DBX 6.3, 453
DBX4.2, 284
DB21, ...
DBB12/16/20, 309
DBB12 - 23, 326
DBX 12.0 - 12.2 ff, 296
DBX0.3, 311, 316, 1040
DBX0.6, 311
DBX100.5, 1052
DBX101.5, 1052
DBX102.5, 1052
DBX12.0-2, 1040
DBX12.0-5, 1043
DBX12.3, 723
DBX12.4, 1041
DBX12.5, 1041
DBX12.6, 1042
DBX12.7, 1042
DBX12.7 and/or DBX12.6ff, 1041
DBX13.6, 1043
DBX16.0-2, 1040
DBX16.0-5, 1043

DBX16.4, 1041
DBX16.5, 1041
DBX16.6, 1042
DBX16.7, 1042
DBX17.6, 1043
DBX20.0-2, 1040
DBX20.0-5, 1043
DBX20.4, 1041
DBX20.5, 1041
DBX20.6, 1042
DBX20.7, 1042
DBX21.6, 1043
DBX24.3, 316, 1045
DBX3.0, 723, 1068
DBX3.1, 723, 1069
DBX3.2, 723, 1069
DBX3.3, 1069
DBX3.4, 723, 1070
DBX3.5, 1070
DBX30.0, 1050
DBX30.0, 30.1, 30.2, 313
DBX30.1, 1050
DBX30.2, 1050
DBX30.3, 313, 1051
DBX30.4, 314, 1051
DBX33.3, 308, 1045
DBX33.6, 1063
DBX332.4, 1048
DBX332.5, 1048
DBX332.6, 1049
DBX332.7, 1049
DBX336.4, 1048
DBX336.5, 1048
DBX336.6, 1049
DBX336.7, 1049
DBX340.4, 1048
DBX340.5, 1048
DBX340.6, 1049
DBX340.7, 1049
DBX37.0, 1051
DBX37.0-2, 1046
DBX37.1, 1051
DBX37.2, 1051
DBX38.0, 724, 1070
DBX38.1, 723, 1071
DBX40.0-2, 1046
DBX40.4, 1046
DBX40.5, 1046
DBX40.6, 1047
DBX40.7, 289, 295, 1047
DBX41.0-6, 1048
DBX46.0-2, 1046
DBX46.4, 1046
DBX46.5, 1046
DBX46.6, 1047
DBX46.7, 1047
DBX47.0-6, 1048
DBX52.0-2, 1046
DBX52.4, 1046
DBX52.5, 1046
DBX52.6, 1047
DBX52.7, 1047
DBX53.0-6, 1048
DB21, …
DBX 36.5, 453
DBX13.6, ff, 289
DBX40.6, 289, 295
DBX41.6, ff, 289
DB31
DBX2.0, 706
DBX62.0, 706
DB31, ...
DBB0, 1072
DBB5, 1056
DBB5 ff., 292
DBB68, 1062
DBB78-81, 1075
DBB8, 1062
DBX1.7, 285
DBX100.2, 1077
DBX100.3, 1078
DBX100.4, 1078
DBX100.5, 1078
DBX100.6, 1078
DBX100.7, 1078
DBX104.0 - 7, 1079
DBX12.4, 852, 861, 1079
DBX13.0-2, 332, 333, 1057
DBX16.7, 328
DBX17.6, 328
DBX18.5, 328
DBX18.7 or DBX18.6, 328
DBX2.0, 1067
DBX2.2, 293, 297, 306, 307, 1072
DBX28.0, 1076
DBX28.1, 777, 1073
DBX28.2, 777, 1073
DBX28.3, 1076
DBX28.4, 1076
DBX28.5, 1077
DBX28.6, 776, 1077
DBX28.7, 774, 1077
DBX29.5, 905
DBX31.4, 906
DBX31.5, 1080
DBX4.0 - DBX4.2, 296

DBX4.0 to DBX 4.4, 307
DBX4.0-2, 1053
DBX4.4, 1053
DBX4.5, 1054
DBX4.6, 1055
DBX4.7, 1055
DBX5.6, 288, 1056
DBX6.2, 776, 777
DBX60.1, 1035
DBX60.4, 942, 943
DBX60.4/5, 347
DBX60.5, 942, 943
DBX60.6, 778
DBX60.7, 778
DBX60.7 or DBX60.6, 328
DBX61.1, 1035, 1073
DBX61.2, 1035
DBX61.3, 905
DBX62.0, 1067
DBX62.1, 308, 310, 1058
DBX62.3, 1064
DBX63.0, 776, 780, 1074
DBX63.1, 776, 1074
DBX63.2, 776, 778, 780, 1074
DBX64.0-2, 1058
DBX64.4, 1059
DBX64.5, 1059
DBX64.6, 778, 1059
DBX64.6 and 64.7, 308
DBX64.7, 778, 1059
DBX65.0-6, 1060
DBX65.6, 288
DBX74.4, 852, 861, 1079
DBX75.0-2, 332, 1061
DBX75.3-5, 332, 1061
DBX76.5, 1074
DBX76.6, 942, 943, 945, 1083
DBX83.3, 1084
DBX83.5, 328
DBX83.6, 1085
DBX83.7, 328
DBX84.1, 1085
DBX84.4, 1080
DBX98.0, 1081
DBX98.1, 1081
DBX98.2, 1081
DBX98.4, 1082
DBX99.0, 1082
DBX99.1, 1082
DBX99.4, 906
DB31, …
DBX 61.2, 453
Define handwheel 1 as contour handwheel, 1052
Define handwheel 2 as contour handwheel, 1052
Define handwheel 3 as contour handwheel, 1052
Defining geometry axes, 578
Deformation
due to temperature effects, 345
Delayed stroke, 1069
Delete distance-to-go, axis-specific, 1072
Determinism, 925
Differential speed, 913
Digital I/Os|840Di, 29
Direct connection, 92
Direction of rotation, 508, 532
Disable analog NCK inputs, 1018
Disable analog NCK outputs, 1020
Disable digital NCK inputs, 1016
Disable digital NCK outputs, 1016
Disable synchronization, 1080
Double addressing, 90
Dressing during machining process, 991
DRF, 283, 315, 962
Drive control, 101
Drum, 101
Drum/rotary switching axis, 99
Dynamic
HMI property, 89
Switchover, 81
Dynamic NC memory, 925
Dynamic programming in spindle/axis operations, 919
Dynamic response
Adaptation, 405
Dynamic user memory, 932

E

Effects on HMI operation, 554
End-of-motion criterion
Braking ramp IPOBRKA, 793
FINEA, COARSEA, IPOENDA, 792
with block search, 798
Entry of channel axes, 505, 529
Error
Angularity, 364
Leadscrew, 358
Measuring system, 358
Sag, 364
Temperature compensation curves, 345
Error during oscillation movement, 1078
Example, 555
Cont. measurement on completion of progr. traversing motion, 687
Continuous measurements modally over several blocks, 688

Continuous measurements with deletion of distance-to-go, 688
Measuring mode 1, 686
Measuring mode 2, 687
One operator panel: three NCUs, 221
TRAANG, 590
TRACYL, 585
TRANSMIT, 583
Two operator panels : one NCU, 221
Example of functional test, 689
Exceptions, 548
Extensions, 547
External oscillation reversal, 1076

F

Faults, 91
FC18, 799
Feed, 783
Feed override, 783, 794, 798
Feedforward control
Speed, 401
Torque, 403
Feedforward control, 399
Feedrate override, 285
Feedrate override/spindle override axis-specific, 1072
Feedrate/rapid traverse override, 326
FINEA, 792
Fixed point approach, 970
Fixed point positions, 334
Following error, 399
Following spindle
Resynchronization, 906
Following spindle interpolator, 877
Frames, 555
Friction compensation
Conventional, 410
Friction compensation, 409
Friction compensation (quadrant error compensation)
Amplitude adaptation, 410
Characteristic parameters, 411
Commissioning, 412
Quadrant errors, 409
FS (following spindle) active, 1082

G

G75, 331
G751, 331
General channel configuration, 591
General information, 606
General reset, 92
Generic coupling, 875
Geometry axes, 286
Geometry axis grouping are either, 488
Geometry monitoring, 1084
Geometry-axis manual travel, 326
GET, 477
GETD, 477
Global data, 90
Grid structure, 374
Grinding operations, 539
Grinding tools, 976
Groove machining, 524
Groove traversing-section, 526
GWPS, 1006
in all operating modes, 1008
GWPS active, 1085

H

Handwheel
Assignment, 296
Connector, 295
Path definition, 312
Selection of HMI, 296
Traversal in JOG, 295
Velocity specification, 312
Handwheel 1 active as contour handwheel, 1051
Handwheel 2 active as contour handwheel, 1051
Handwheel 3 active as contour handwheel, 1051
Handwheel connection
Ethernet, 322
Handwheel override in AUTOMATIC mode
Path definition, 306
Programming and activation, 309
Velocity override, 306
Handwheel selected (for handwheel 1, 2 or 3), 1038
Hardware limit switches, 328
HHU, 90
Hirth tooth system, 955
HMI
Property static/dynamic, 88
State, 85
Switchover, 118
HW_ASSIGN|840Di, 31

I

I/O device interface, 611
I/O range, 58
Configuration, 59
Configured length, 58
Start address, 58

System variables, 62
Identification
Operator panels, 132
Identification of spindles, 505
INC, 942, 943
INCH or METRIC unit of measurement, 628
Inclined axis
TRAANG, 539
Inclined axis transformations, 582
Incremental travel, 292
Incremental travel (INC), 943
Continuous operation, 292
Jog mode, 292
Indexing axes
Coded position, 948
FRAMES, 952
Handwheel, 943
Programming, 948
Reference point approach, 962
Startup, 959
Indexing axis
Parameter setting, 946
Indexing axis in position, 1083
Indexing positions
Number, 946
Indexing positions table, 946
Infeed, 809
Initial learning, 434
Input override for geometry axis (1, 2, 3), 1041
Input values, 616
Measurement types, 617
Setpoints, 619
Interface, 287, 611
Interface signals
Activate handwheel (1 to 3), 1053
Active machine function INC1, ..., continuous, 1060
Active machine function for geometry axis (1, 2, 3), 1048
Continuous machine function, 1056
Contour handwheel active (1 to 3), 1046
DRF selected, 1045
Handwheel active (1 to 3), 1058
Handwheel active (1 to 3) for geometry axis, 1046
Handwheel override active, 1045, 1058
JOG - Approaching fixed point 0/1/2, 1057
JOG - Approaching fixed point active, 1061
JOG - Approaching fixed point reached, 1061
Machine function INC1, INC10, INC100, INC1000, INC10000, INCvar, 1056
Plus and minus travel command, 1059
Plus and minus travel command (for orientation axis), 1049
Plus and minus travel commands (for geometry axis), 1047
Plus and minus traverse keys, 1055
Plus and minus traversing request, 1059
Plus and minus traversing request (for geometry axis), 1046
Plus and minus traversing request (for orientation axis), 1048
Rapid traverse override, 1054
Traversing-key lock, 1053
Interpolation, 101
Linear, 771
Non-linear, 772
with G0, 771
Interpolation cycle, 103
Interpolation functions, 805
Interpolation point, 355
Interpolator
Path, 771
Shaft, 771
Interrupt parts program, 513, 538
IPOBRKA, 793, 795
IPOENDA, 792
IS Feedrate stop/Spindle stop (DB31, ... DBX4.3), 888

J

JOG, 330, 555, 942
Approaching a fixed point, 331
JOG mode, 284
Jogging operation, 942

L

Language command
SPN, 736, 738
SPP, 735, 737
Language commands, 725
Large batch production, 99
Learning ON / OFF, 431
Learning the neural network, 429
Link
Axis, 101, 102
Link axes
Typical link cycles, 144
Link function, 144
Link module, 102, 103
Link variables, 81
Global, 103
Linked transformation
Example, 552
Local NCU, 96

Longitudinal grooves, 498
LS (leading spindle) active, 1082

M

M cabling
N, 88
Machine
Control panel, 86
State, global, 103
Machine axis (for handwheel 1, 2 or 3), 1038
Machine axis identifier, 390
Machine control panel, 287
Machine function continuous for geometry axis (1, 2, 3), 1043
Machine function for geometry axis (1, 2, 3), 1043
Main
Control panel, 81
Secondary control panel, 85
Manual stroke initiation, 1069, 1070
Manual travel, 326
Manual travel in JOG, 284
Master, slave communication, 80
MCP switchover, 124
MCP switchover disable, 1032
MD10000, 476
MD10002, 145, 147, 148, 156, 161
MD10010, 461
MD10065, 192, 193, 194
MD10087, 245
MD10088, 454
MD10134, 91
MD10200, 420
MD10210, 420, 866
MD10260, 196, 363, 371, 703
MD10270, 703, 947
MD10300, 30, 32, 43, 44
MD10310, 30, 32, 45
MD10320, 33, 44
MD10330, 33, 47
MD10350, 30, 32, 35, 41
MD10360, 30, 32, 37, 39, 41
MD10361, 42
MD10362, 30
MD10364, 30
MD10366, 30
MD10368, 30
MD10382, 34
MD10384, 34, 44
MD10394, 50
MD10395, 50
MD10396, 50
MD10397, 50
MD10398, 50, 69
MD10399, 51
MD10450, 704
MD10460, 704, 713
MD10461, 704, 713
MD10470, 698, 707, 709, 711
MD10471, 707, 709, 711
MD10472, 707, 709, 711
MD10473, 698, 707
MD1048, 710
MD10480, 708, 711
MD10485, 702, 709, 710, 711
MD10500, 59
MD10501, 59
MD10502, 60
MD10510, 59
MD10512, 60, 64
MD10530, 54, 55
MD10531, 55
MD10540, 55
MD10722, 483, 484, 485, 800
MD10735, 334
MD10900, 946, 958, 960
MD10910, 946, 958, 959, 960
MD10920, 946, 958, 959, 961
MD10930, 946, 958, 961
MD10940, 949, 952
MD11300, 292
MD11310, 298
MD11322, 312
MD11330, 292, 307
MD11346, 300, 301, 302, 312, 333
MD11350, 319, 320
MD11351, 319, 320
MD11352, 319, 320
MD11353, 320
MD11410, 453
MD11450, 60, 733
MD12510, 174, 175, 176, 183
MD12520, 176
MD12530, 175
MD12540, 175
MD12550, 175
MD12701, 156, 165, 255
MD12702, 165, 255
MD12703, 165, 255
MD12704, 165, 255
MD12705, 165, 255
MD12706, 165, 255
MD12707, 165, 255
MD12708, 165, 255
MD12709, 165, 255

MD12710, 165, 255
MD12711, 165, 255
MD12712, 165, 255
MD12713, 165, 255
MD12714, 165, 255
MD12715, 165, 255
MD12716, 165, 255
MD12717, 165
MD12750, 156, 160
MD13210, 612
MD1500, 443
MD18050, 933
MD18060, 930, 931
MD18096, 976, 979, 981
MD18100, 981
MD18210, 933
MD18230, 929
MD18342, 357, 368
MD18351, 664
MD18352, 929, 931
MD18353, 929, 931
MD18402, 183
MD18404, 183
MD18600, 620
MD18720, 183, 191
MD18780, 142, 168, 175, 183, 195
MD18781, 177, 178
MD18782, 183
MD20000, 92
MD20050, 851
MD20070, 146, 156, 160, 471, 472, 476
MD20100, 330, 629
MD20110, 196, 471, 747, 1008
MD20112, 471
MD20120, 1008
MD20130, 1008
MD20150, 629, 721
MD20254, 1001
MD20350, 980, 1002
MD20360, 629
MD20390, 348
MD20610, 1000
MD20620, 297
MD20624, 300, 303
MD20730, 772
MD20750, 772
MD21106, 568
MD21150, 867
MD21220, 33, 56
MD22550, 969
MD22560, 969
MD26000, 729
MD26002, 729
MD26004, 729
MD26006, 729
MD26010, 736, 741
MD26014, 735, 736, 745
MD26016, 741
MD26018, 721, 730
MD26020, 724, 730
MD30300, 849, 861, 866, 947, 959
MD30310, 850, 852, 854, 855, 857, 861, 866, 947, 959
MD30320, 852, 854, 857, 866
MD30330, 858, 950
MD30340, 855, 856, 858
MD30460, 799, 803
MD30500, 294, 946, 955, 958, 961
MD30503, 856
MD30505, 955
MD30550, 160, 161, 471, 472, 476, 478
MD30552, 470, 471, 478, 769
MD30554, 176, 183
MD30560, 176
MD30600, 334
MD30610, 334
MD31090, 292, 295, 307, 312
MD32000, 308, 771, 772
MD32010, 285
MD32020, 285, 297, 327
MD32040, 285, 299, 330, 783, 944
MD32050, 285, 299, 330, 783, 944
MD32060, 311, 770, 783, 789, 791, 801
MD32074, 488
MD32080, 297, 298
MD32084, 300, 303
MD32090, 315
MD32300, 329
MD32420, 298, 772
MD32430, 772
MD32450, 353
MD32452, 354
MD32490, 409, 410, 423, 434, 443
MD32500, 413, 422, 423, 435, 443
MD32510, 410, 413
MD32520, 411, 413
MD32530, 411
MD32540, 413, 425
MD32550, 411, 419, 420
MD32560, 412, 420
MD32570, 412, 420
MD32580, 420, 423, 442, 443
MD32610, 402, 443
MD32620, 399, 433
MD32620, 443
MD32630, 400, 443
MD32650, 404

MD32700, 356, 359
MD32710, 356, 366
MD32711, 372
MD32720, 372
MD32730, 372
MD32750, 348, 350
MD32760, 349
MD32800, 404
MD32810, 400, 401, 443
MD32900, 406
MD32910, 405
MD32990, 193, 194
MD35032, 1008, 1009
MD35040, 1009
MD36100, 861
MD36110, 861
MD36500, 354
MD36610, 453
MD36620, 453
MD37500, 781, 782
MD37510, 781, 782
MD37511, 781, 782
MD38000, 357, 360
MD38010, 423, 434, 438, 443, 444
Measurement
  of angle in a plane ($AC_MEAS_TYPE = 17), 649
  of groove ($AC_MEAS_TYPE = 12), 641
  of hole ($AC_MEAS_TYPE = 8), 637
  of oblique edge ($AC_MEAS_TYPE = 16), 647
  of shaft ($AC_MEAS_TYPE = 9), 640
  of web ($AC_MEAS_TYPE = 13), 644
Measurement accuracy, 685
Measurement input parameters, 622
Measurement interface
  Diagnostics, 629
  Input values, 617
  Output values, 624
Measurement method
  for coordinate transformation of a position ($AC_MEAS_TYPE = 24), 659
  For defining an additive rotation of the active or selected plane ($AC_MEAS_TYPE = 28), 666
  For determining a triangle ($AC_MEAS_TYPE = 25), 663
  For restoring the value assignments of data management frames ($AC_MEAS_TYPE = 27), 665
  For saving data management frames with current value assignments to a file ($AC_MEAS_TYPE = 26), 664
Measurement of tool diameter ($AC_MEAS_TYPE = 11), 670
Measuring cycles, 626
Measuring probe connection, 608
Measuring status, 1064
Measuring system, 947
Mechanical backlash, 352
Memory expansion, 930
Memory organization, 925
Menu
  Connections/Service, 92
Milling/drilling unit, 99
Minimum interval between two consecutive strokes, 730
Minus cam signals 1-32, 1066
MIRROR(C), 868
Mode group, 459, 461
Modes, 330
Modified activation of machine data, 958
Modular machine concept, 79
Modulo 360, 854
Modulo rotary axis, 947
  Working-area limitation, 861
Monitoring functions, 328
Monitoring of the input signal, 730
Monitoring status with modulo rotary axes, 1079
Monodirectional probe, 607
Motion behavior, 805
Motion control, 101
MPI, 86, 90
MPI, network rules, 97
Multidirectional probe (3D), 606
Multi-face machining, 99
Multiplication
  Table, 366
Multi-point interface (MPI), 86
Multi-spindle turning machine, 99

N

NC
  Address, 87, 90
NCK digital inputs, 35
NCK I/O via PROFIBUS
  Activation, 59
NCK I/Os, 28
NCK treats the axis as a positioning axis, 1074
NCU
  Link, 85, 98
  Replacement, 92
NCU link, 389
NCU link active, 1034
NCU link axis active, 1035
NCU-NCU communication, 98
NETNAMES.INI, 89, 91
  Syntax, 132

Network rules, 97
Networked NCUs, 98
Neural quadrant error compensation, 421
  Commissioning, 434
  Optimize, 437
  Parameterization, 423
Neutral axis, 470
No stroke enable, 1068
Not transformation-specific, 582
Number of bus nodes, 90
Number of chained transformations, 551
Number of inclined axes, 541
Number of transformations, 503, 526, 551

O

OEM solution, 95, 96
Offline
  Requirement, 118
OP030, 90
Operating
  Area, 91, 96
  Display, 95, 96
  Unit, 79, 94
Operating mode changeover rejected, 1033
Operating mode switchover, 124
Operator panel, 79
Operator panel front, 86
Operator panel interface (OPI), 86
OPI, 86, 90
OPI default, 87
OPI, network rules, 97
Optimization of velocity control, 545
Orientation, 571
Orientation in TCS, 573
Orientation transformations, 580
OS, 815
OSB, 817
OSCILL, 823
Oscillating, 809
  Asynchronous, 809
  Continuous infeed, 809
  with synchronized actions, 835
Oscillating axis, 809
Oscillation active, 1078
Oscillation cannot start, 1078
Oscillation movement active, 1078
Oscillation reversal active, 1077
OSCTRL, 816, 817
OSE, 817
OSNSC, 817
OSP, 815
OST, 816
Output values, 616
Overlap areas of axis angles
  TU address, 565
Overwrite mask for analog NCK outputs, 1019
Overwrite mask for digital NCK outputs, 1017

P

Part program, 102, 103, 594
Passive file system, 926
Path
  Interpolator, 771
Path axes
  Path interpolator, 788
Path default using handwheel, 312
Path segmentation, 735
Permanent coupling configuration, 877
Permissible angular range, 544
PG diagnostics, 87
Plane separation, 619
PLC
  Address, 87, 90
  Local I/Os, 80
  PLC communication, 80, 82
PLC axes, 799
PLC axis, 470, 481
  Axes under exclusive PLC control, 799
  Exclusively PLC-controlled, 486
  Permanently assigned, 487
  Permanently assigned PLC axis, 799
  Start via FC18, 802
PLC controls axis, 1077
PLC service display, 614
PLC-controlled axis, 1074
  Control response to MD30460 bits 6 and 7, 803
Plus and minus traverse keys for geometry axis (1, 2, 3), 1042
Plus cam signals 1-32, 1066
Position control, 101
Position of tool zero, 508, 533
Position offset
  In synchronous spindles, 905
Positioning axes, 764
  Acceleration values, 783
  Axis interpolator, 788
  Axis types, 768
  Axis-specific signals, 799
  Block change, 788
  Channel-specific signals, 799
  Concurrent, 769, 799
  Dry run feedrate, 804

Maximum number, 784
Path interpolator, 793
Positioning axis type 1, 788, 789
Positioning axis type 2, 788, 790
Positioning axis type 3, 788, 792
Tool offset, 786
Types, 767
Velocity values, 783
Position-switching signals, 693
Position-time cams, 712
Features, 712
POSP, 823
Power On, 561
Preprocessing stop, 61
Preset actual value memory, 615
for geo axes and special axes ($AC MEAS TYPE = 14), 645
for special axes ($AC MEAS TYPE = 15), 646
Probe actuated, 1064
Probe assignment, 606
Probe types, 606
Probes that can be used, 606
Processing
Face-end, 497
PROFIBUS, 90, 102
Interface, 98
Program coordination, 462
Example, 464
Programming of joint position
STAT address, 564
Programming variants, 546
Programming with groove wall offset, 586
Programming without groove wall offset, 588
Protection level, 94
Protection level Service, 92
Protocol layer, 84
PTP/CP switchover
Mode change in JOG, 567
PUNCHACC, 731

Q

Quadrant error compensation, 409, 421
Quantization of characteristic, 427
Quick commissioning, 443

R

RangeIndex, 63
RangeOffset, 63
Rapid traverse
Interpolation types, 771
Rapid traverse override, 285, 783
Read measurement results in PP, 614
Read offset, 888
Reciprocating axes, 793
Reconfiguration, 925, 927
Redefine WCS on the oblique plane ($AC_MEAS_TYPE = 18), 652
Reference point approach, 317, 942
Reference point setting in relative coordinate systems, 645
Relearning, 435
RELEASE, 476
Release gantry master axis, 476
Replaceable geometry axes, 533, 545
Replaceable geometry axis, 509
Reset, 562
Reset axis/spindle, 1073
Response to setpoint changes, 916
Restart, 91
Resume axis/spindle motion, 1073
Resynchronization, 905
Reversal points, 809
Rotary
Axis, 99
Rotary axes, 849
Absolute programming, 858, 863
Activation, 866
Axis addresses, 850
Feedrate, 852
Incremental programming, 862, 865
Mirroring, 868
Modulo 360, 854
Modulo conversion, 857, 863
Operating range, 852
Positioning display, 852
Software limit switch, 868
Units of measurement, 851
Rotary axis, 849
Rotary indexing machine, 99
Rotational position, 507, 532
RTLIOF, 773
RTLION, 773
Runtime, 89

S

Sag compensation, 389
Scratching, 615
SD41010, 292, 297, 298, 307
SD41040, 942
SD41050, 289, 942
SD41100, 285, 299, 330, 783, 853, 944

SD41110, 285, 297, 327
SD41120, 285, 330
SD41130, 285, 297, 867
SD41200, 327
SD41300, 366
SD41500, 702
SD41500 - 41507, 947
SD41501, 702
SD41502, 702
SD41503, 702
SD41504, 702
SD41505, 702
SD41506, 702
SD41507, 702
SD41520, 705, 713
SD41521, 705, 713
SD41522, 705
SD41523, 705
SD41524, 705
SD41525, 705
SD41526, 705, 713
SD41527, 705, 713
SD41600, 56
SD41700, 158, 159, 160
SD42100, 311, 804, 812
SD42101, 804, 812
SD42400, 727
SD42402, 721, 730
SD42404, 730
SD42600, 299, 783, 787, 944
SD43300, 299, 783, 787, 944
SD43400, 861
SD43410, 861
SD43600, 793, 795, 796
SD43610, 795
SD43770, 817
SD43790, 817
SD43900, 348
SD43910, 348
SD43920, 348, 350
Second operator panel, 93
Secondary
  Control panel, 81
Secondary conditions, 562
Secondary control panel, 85
Selecting reference systems, 569
Selection, 562
Selection and deselection, 547, 553
Selection of tool or cutting edge, 622
Separate following spindle interpolator, 877
Series machine start-up file, 927
Service case, 92
Service/commissioning, 87
Set reversal point, 1076
Set-change criteria IPOBRKA
  WAITMC occurs, 466
SETM, 463
Setpoint for analog NCK outputs, 1024
Setpoint for digital NCK outputs, 1023
Setpoints, 619
Settable block change time, 792
Setting mask for analog NCK inputs, 1018
Setting mask for analog NCK outputs, 1019
Setting mask for digital NCK outputs, 1018
Setting on PLC of digital NCK inputs, 1016
Setting value from PLC for analog NCK inputs, 1019
Setting value from PLC for analog NCK outputs, 1020
Setting value from PLC for the digital NCK outputs, 1017
Several NCUs, 94, 268
Several operator panels
  Alarms/Messages, 138
  Buses, 133
  Compatibility, 115
  Configurations, 131
  Connections, 132
  Defaults, 115, 135
  Implementation, 93
  NCU components, 134
  Operating characteristics, 94
  Operational characteristics, 280
  Operator components, 133
  Operator interface, 138
  Operator interfaces, 268
  Switchover of connection, 135
Single axes
  Alarm handling, 777
  Applications, 776
  Axis control by PLC, 774
  Extended retract numerically controlled, 781
  Extended stop numerically controlled, 781
Single block
  Positioning axis type 1, 804
  Positioning axis type 2, 804
  Positioning axis type 3, 804
Single transformations, 592
Slide, 99
Slot side compensation, 499
Softkey, 91, 96
Software cam, 693
Software limit switch, 328, 868, 962
Sparking-out active, 1078
Sparking-out strokes, 809
Special features, 563
Special features of JOG, 538
Speed monitoring, 1004, 1085

Spindle manual travel, 327
Spindle number, 982
Spindle position with mono probe, 607
Spindle replacement, 470
Standard alarm texts, 92
Start operating area, 91
Static HMI property, 88
Static NC memory, 925
Static user memory, 928
Station/position change, 102
Status query
HW outputs, 708
Stop along braking ramp, 1077
Stop at next reversal point, 1077
Stroke initiation active, 1070
Stroke inoperative, 1070
Stroke suppression, 1069
Superimposed motion, 1082
Suppression
Algorithm, 89, 118
Mechanism, 85
Rules, 119
Strategy, 89, 119
Switchover
Attempt, 95
Behavior on OP030, 91
Conditions, 120
Time, 91
Symbol name, 89
Synchronism coarse, 1081
Synchronism fine, 1081
Synchronized action, 103
Synchronized state reached, 887
Synchronous mode, 876, 1080
Deactivate, 896
Knee-shaped acceleration characteristic, 919
Synchronous spindle
Position offset, 905
System variables, 103, 554, 614
Global, 103

T

Table
Compensation, 355, 356
Task specification, 524, 539
Temperature
Compensation, 345
Influence, 345
Temperature compensation
Coefficient tanß(T), 351
Temporary assignment, 81
Terminal X143, 318
Test program for testing repeat accuracy, 689
Time constant
Dynamic response adaptation, 405
Tool change
Fixed points, 970
Sequence, 967
Tool change point, 970
Tool change times, 968
Tool length
($AC_MEAS_TYPE = 10), 668
Measurement with stored or current position ($AC_MEAS_TYPE = 23), 672
Measurement with zoom-in function ($AC_MEAS_TYPE = 22), 671
Tool measurement of two milling tools
Each with their own reference point, 677
With one reference point, 678
Tool measurement of two turning tools
Each with their own reference point, 674
With one reference point, 675
Tool measuring, 667
of two milling tools each with their own reference point, 680
of two milling tools with one reference point, 681
Two turning tools each with their own reference point, 673
Tool offset for grinding tools, 976
Tool types for grinding tools, 980
TRAANG
Activating, 546
Amount, 541
Brief description, 500
Deactivation, 546
Inclined axis, 539
Restrictions, 547
Specific settings, 543
TRAANG_Angle_m, 544
TRAANG_BASE_TOOL_m, 544
TRAANG_PARALLEL_ACCEL_RES_m, 545
TRACYL, 498
Amount, 527
Axis image, 531
Restrictions, 536
Tracyl transformations, 581
TRACYL_BAE_TOOL_t, 533
TRACYL_ROT_AX_OFFSET_t, 532
TRACYL_Rot_Sign_IS_PLUS_t, 532
TRAFO_AXES_IN_n, 507
TRAFO_TYPE, 530
TRAFO_TYPE_n, 506, 543
TRANS, 865

Transformation
Chaining sequence, 551
Transformation active, 1063
Transformation chain, setpoint positions, 558
Transformation type 257, 506
Transitions of possible axis states during axis replacement, 475
Translation, 569
Translation and orientation in the TCS simultaneously, 571
Translation in the BCS, 569
Translation in the TCS, 570
Translation in the WCS, 570
Translational offsets, 619
TRANSMIT, 497, 502
Activating, 510
Amount, 503
Axis image, 506
Deactivation, 510
Restrictions, 511
Specific settings, 506
Transmit transformations, 581
TRANSMIT_ROT_AX_OFFSET_t, 507
Transverse axes, 330
Transverse grooves, 498
Traversal from PLC, 945
Traverse key disable for geometry axis (1, 2, 3), 1041
Traversing range limitation for modulo rotary axes, 1079
Type of transformation, 530, 543

U

User
Alarm, 92
User communication, 81
User-defined coupling, 877
Utilization property, 88

V

Variable interface, 616
Velocity, 285, 297, 327
Velocity and acceleration, 329
Velocity control, 513, 548

W

WAITE, 463
WAITM, 463
WAITMC, 463, 465
and read-in disabled, 468
and SETM, 466
WAITP, 786
Oscillating axis, 823
Working-area limitation, 328, 868
Workpiece clamping, 99, 103
Workpiece measuring, 616
Write online tool offset discretely, 998

X

x edge ($AC_MEAS_TYPE = 1), 630
x edge measurement, 631
X143, 318

Y

y edge ($AC_MEAS_TYPE = 2), 632

Z

z edge ($AC_MEAS_TYPE = 3), 633