

## DOCUMENTATION ISG-kernel

## Manual <br> Programming manual

## Short Description: PROG

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
All rights reserved
www.isg-stuttgart.de
support@isg-stuttgart.de

## Preface

## Legal information

This documentation was produced with utmost care. The products and scope of functions described are under continuous development. We reserve the right to revise and amend the documentation at any time and without prior notice.

No claims may be made for products which have already been delivered if such claims are based on the specifications, figures and descriptions contained in this documentation.

## Personnel qualifications

This description is solely intended for skilled technicians who were trained in control, automation and drive systems and who are familiar with the applicable standards, the relevant documentation and the machining application.

It is absolutely vital to refer to this documentation, the instructions below and the explanations to carry out installation and commissioning work. Skilled technicians are under the obligation to use the documentation duly published for every installation and commissioning operation.

Skilled technicians must ensure that the application or use of the products described fulfil all safety requirements including all applicable laws, regulations, provisions and standards.

## Further information

This link
https://www.isg-stuttgart.de/de/isg-kernel/kernel-downloads.html
contains further information on messages generated in the NC kernel, online help, PLC libraries, tools, etc. in addition to the current documentation.

## Disclaimer

It is forbidden to make any changes to the software configuration which are not contained in the options described in this documentation.

## Trade marks and patents

The name ISG $^{\circledR}$, ISG kernel ${ }^{\circledR}$, ISG virtuos ${ }^{\circledR}$, ISG dirigent ${ }^{\circledR}$ and the associated logos are registered and licensed trade marks of ISG Industrielle Steuerungstechnik GmbH.
The use of other trade marks or logos contained in this documentation by third parties may result in a violation of the rights of the respective trade mark owners.

## Copyright

© ISG Industrielle Steuerungstechnik GmbH, Stuttgart, Germany.
No parts of this document may be reproduced, transmitted or exploited in any form without prior consent. Non-compliance may result in liability for damages. All rights reserved with regard to the registration of patents, utility models or industrial designs.

## General and safety instructions

## Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

## Icons in explanatory text

Indicates an action.
$\Rightarrow$ Indicates an action statement.


## 4. DANGER

## Acute danger to life!

If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.


## 4 CAUTION

## Personal injury and damage to machines!

If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

## Attention

## Restriction or error

This icon describes restrictions or warns of errors.


## Notice

## Tips and other notes

This icon indicates information to assist in general understanding or to provide additional information.


## Example

## General example

Example that clarifies the text.

## Programing Example

## NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

## Release Note

## Specific version information

Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

## Table of contents

Preface ..... 2
General and safety instructions ..... 3
1 Brief description ..... 19
2 Basic principles of programming ..... 20
2.1 Typographical representation of syntax ..... 20
2.2 Character and number formats ..... 20
2.2.1 Character set and file format. ..... 20
2.2.2 Numerical input ..... 21
2.3 Structure of NC control data: NC programs ..... 23
2.4 NC block structure ..... 24
2.4.1 Skipping NC blocks '/' ..... 26
2.4.1.1 Standard skipping ..... 26
2.4.1.2 Extended skipping (skip levels) ..... 26
2.4.2 Block-specific comments ..... 27
2.4.3 Line break in NC block 'l' ..... 28
2.5 Word structure ..... 30
2.5.1 Mathematical expressions ..... 30
2.5.1.1 Integers <int> ..... 30
2.5.1.2 Decimal numbers <double> ..... 30
2.5.1.3 Arithmetic expressions <expr> ..... 31
2.5.2 Operations for character strings. ..... 36
2.5.3 Assigned address characters ..... 41
2.5.4 Programming examples ..... 43
3 Path information ..... 44
3.1 Axis commands ..... 44
3.2 Measuring systems, input and precision ranges ..... 46
3.3 Coordinate systems ..... 47
4 G functions ..... 52
4.1 Path preparatory functions ..... 53
4.1.1 Rapid traverse (G00) ..... 53
4.1.2 Linear interpolation (G01) ..... 54
4.1.3 Circular interpolation (G02/G03) ..... 55
4.1.4 Helical interpolation (G02 Z.. K../G03 Z.. K..) ..... 59
4.1.4.1 Simple helical interpolation ..... 65
4.1.5 Arc in space (G303) ..... 66
4.1.6 Contour line programming (\#ANG) ..... 68
4.1.7 Dwell time (G04), (\#TIME) ..... 83
4.1.8 Programmable homing (G74) ..... 84
4.1.9 Reference point offset (G92). ..... 85
4.1.10 Set negative software limit switch (G98) ..... 86
4.1.11 Set the positive software limit switch (G99) ..... 88
4.1.12 Extensions to G98 and G99 ..... 90
4.1.13 Measuring functions ..... 90
4.1.13.1 Measure with multiple axes (G100) (Type 1) ..... 92
4.1.13.2 Measure with a single axis (G100) (Type 2) ..... 94
4.1.13.3 Measure with motion through to target point (G100/G106) (Type 3) ..... 96
4.1.13.4 Measure with main axes (G100) (Type 4) ..... 98
4.1.13.5 Measure with interruption and jump (G310) (Types 5, 6) ..... 100
4.1.13.6 Measure with motion to a fixed stop (G100) (Type 7) ..... 101
4.1.13.7 Calculate measuring offsets (G101/G102) ..... 102
4.1.13.8 Edge banding (G108) ..... 104
4.1.13.8.1 Glue in one motion block (Method 1) ..... 106
4.1.13.8.2 Edge banding across several motion blocks (G107)(Method 2) ..... 107
4.1.13.8.3 Program distance to go ..... 108
4.2 Determining acceleration/deceleration (G08/G09/G900/G901) ..... 109
4.3 Path/time-related feed interpolation (G193/G293) ..... 112
4.4 Selection of planes (G17/G18/G19) ..... 114
4.5 Mirroring in the plane (G21/G22/G23/G20) ..... 115
4.6 Mirroring with axis specification (G351) ..... 120
4.7 Units (G70/G71) ..... 123
4.8 Implicit subroutine calls (G80-G89/G800..) ..... 123
4.9 Dimension systems (absolute dimension/incremental dimension) (G90/G91) ..... 124
4.9.1 Exclusive programming ..... 124
4.9.2 Combined programming ..... 125
4.10 Exact stop (G60/G360/G359) ..... 126
4.11 Polynomial contouring (G61/G261/G260) ..... 127
4.12 Corner deceleration ..... 129
4.12.1 Parameterising corner deceleration (\#CORNER PARAM) ..... 130
4.12.2 Selecting/deselecting corner deceleration (G12/G13) ..... 131
4.13 Zero offsets (G53/G54/...G59) ..... 132
4.13.1 Enhanced zero offset variables ..... 133
4.13.2 Adding and subtracting offsets ..... 134
4.13.3 Access to the current zero offset ..... 135
4.13.4 Default zero offset ..... 135
4.13.5 Creating zero offset groups ..... 136
4.13.6 Extended zero offset (G159) ..... 137
4.13.7 Enable/disable zero offsets axis-specific (G160) ..... 138
4.14 Specifying centre point for circle definition (G161/G162) ..... 139
4.15 Radius programming (R/G163) ..... 140
4.16 Controlling centre point offset in circle (G164/G165) ..... 144
4.16.1 Special function: circle radius compensation in combination with G164 ..... 146
4.17 Feedforward control (G135/G136/G137) ..... 148
4.18 Weighting of maximum velocity ( $\mathrm{G} 127 / \mathrm{G} 128$ ) ..... 149
4.19 Weighting of rapid traverse velocity (G129) ..... 149
4.20 Parameterising the acceleration profile ..... 150
4.20.1 Acceleration weighting (G130/G131/G231/G333/G334) ..... 150
4.20.2 Ramp time weighting (G132/G133/G134/G233/G338/G339) ..... 153
4.21 Machining time or feedrate (G93/G94/G95/G194) ..... 157
4.22 Inserting chamfers and roundings (G301/G302) (\#FRC/\#CHR/\#CHF/\#RND) ..... 158
4.22.1 Insert chamfers using G301 as example ..... 164
4.22.2 Inserting roundings using G302 as example ..... 166
4.23 Manual mode ..... 168
4.23.1 Selecting/deselecting manual mode with parallel interpolation (G201/G202) ..... 169
4.23.2 Selecting manual mode without parallel interpolation (G200). ..... 171
4.23.3 Reaction at program end (M02, M30) ..... 172
4.23.4 Parameterising operating modes ..... 172
4.23.4.1 Handwheel mode (\#HANDWHEEL) ..... 172
4.23.4.2 Continuous jog mode (\#JOG CONT) ..... 173
4.23.4.3 Incremental jog or interruptible jog mode (\# JOG INCR) ..... 174
4.23.5 Specify offset limits (\#MANUAL LIMITS) ..... 174
4.23.6 Example of parameterising an axis in manual mode ..... 176
4.24 Requesting offset, command and actual values ..... 177
4.24.1 Request current manual mode offsets and file to "V.A.MANUAL_OFFSETS[ ]" (\#GET ..... 178 MANUAL OFFSETS)
4.24.2 Request current command positions and file to "V.A.ABS[ ]" (\#CHANNEL INIT) ..... 179
4.24.3 Request current actual positions and file to "V.A.ABS[ ]" (\#CHANNEL INIT) ..... 181
4.24.4 Request current command positions of axes and file to variables or parameters (\#GET ..... 182 CMDPOS)
4.24.5 Request current actual positions of axes and file to variables or parameters (\#GET ACT- ..... 183POS)
4.25 Gear change (G112) ..... 184
4.26 Influence on the look-ahead functionality (G115/G116/G117) ..... 185
4.27 Override (G166) ..... 188
4.28 Cycle synchronisation at block end (G66) ..... 189
4.29 Rotate the coordinate system in the plane (G68/G69) ..... 190
5 Switching and supplementary functions (M/H/T) ..... 193
5.1 User-specific M/H functions ..... 193
5.1.1 Programmed stop (M00) ..... 194
5.1.2 Optional stop (M01) ..... 194
5.1.3 Program end (M02/M30) ..... 194
5.1.4 Subroutine end (M17/M29) ..... 194
5.1.5 Call a tool change program (M06) ..... 195
5.2 Axis-specific M/H functions ..... 195
5.3 M/H functions with optional additional information ..... 198
5.4 Tool position selection (T ) ..... 199
6 Velocities (F/E) ..... 200
7 NC block numbers (N) ..... 203
8 Subroutine techniques ..... 204
8.1 Local subroutines (Call LL <string>) <string>) ..... 205
8.2 Global sub-routines (Call L <string>) ..... 206
8.3 Parametric subroutine call (LL / L V.E. or macro) ..... 207
8.4 Implicit global subroutine call at program start ..... 210
8.5 Implicit global subroutine call at program end ..... 210
8.6 Cycles as global or local subroutines (Call L | LL CYCLE) ..... 210
8.7 Calling block sequences (L SEQUENCE) ..... 216
9 Parameters and parameter calculation (P) ..... 224
9.1 Programming of coordinates by parameters ..... 227
9.2 Indirect parameters ..... 228
10 Statements for influencing NC program flow ..... 230
10.1 Conditional jumps ..... 232
10.1.1 The IF - ELSE branch ..... 232
10.1.2 Switch branching (\$SWITCH ) ..... 235
10.1.3 The \$GOTO statement ..... 236
10.1.3.1 Parametric jump call ..... 240
10.2 Counting loop (\$FOR) ..... 241
10.3 Loops with running condition. ..... 243
10.3.1 Verification of running condition at loop start (\$WHILE) ..... 243
10.3.2 Verification of running condition at loop end (\$DO), (\$REPEAT) ..... 243
10.4 Influencing loop flow sequences ..... 245
10.4.1 The \$BREAK statement. ..... 245
10.4.2 The \$CONTINUE statement ..... 246
11 Smoothing methods ..... 247
11.1 Programs with several short blocks ..... 248
11.1.1 Trim a contour (\#HSC ON/OFF) ..... 250
11.1.2 Surface machining with Surface Optimiser (method 3) ..... 253
11.1.3 FIR filter (\#FILTER). ..... 257
11.2 Polynomial contouring for long blocks (G61/G261/G260) ..... 258
11.2.1 Definition of terms ..... 259
11.2.2 General properties ..... 261
11.2.2.1 Maximum corner distance, minimum residual block length ..... 261
11.2.2.2 Relevant block length ..... 262
11.2.2.3 Executing additional blocks ..... 265
11.2.2.4 Jerk within the polynomial ..... 266
11.2.2.5 Velocity curve in the contouring section ..... 267
11.2.3 Parameterising contouring modes in the NC program (\#CONTOUR MODE) ..... 268
11.2.4 Activating contouring modes in the NC program ..... 269
11.2.4.1 Contouring with corner deviation ..... 269
11.2.4.2 Corner distance contouring ..... 272
11.2.4.3 Dynamic optimised contouring ..... 274
11.2.4.4 Dynamic optimised contouring with master axis ..... 277
11.2.4.5 Contour with interim point ..... 278
11.2.4.6 Dynamically optimised contouring of the complete contour. ..... 280
11.2.5 Example ..... 284
11.2.6 Remarks ..... 290
11.3 Other processes ..... 291
11.3.1 Akima spline interpolation ..... 291
11.3.1.1 Selecting AKIMA spline type (\#SPLINE TYPE AKIMA ) ..... 291
11.3.1.2 Selecting Akima spline interpolation (\#SPLINE ON) ..... 291
11.3.1.3 Deselecting Akima spline interpolation (\#SPLINE OFF) ..... 292
11.3.1.4 Specifying transition type (spline curve) (\#AKIMA TRANS) ..... 293
11.3.1.5 Defining the start tangent (\#AKIMA STARTVECTOR) ..... 294
11.3.1.6 Defining the end tangent (\#AKIMA ENDVECTOR) ..... 294
11.3.2 B spline interpolation ..... 297
11.3.2.1 Selecting B spline type (\#SPLINE TYPE BSPLINE) ..... 297
11.3.2.2 Selecting B spline interpolation (\#SPLINE ON) ..... 297
11.3.2.3 Deselecting B spline interpolation (\#SPLINE OFF) ..... 298
11.3.3 PSC programming with OP1 and OP2 ..... 300
11.3.3.1 Available operation modes ..... 301
11.3.3.2 Additional parameters ..... 304
12 Additional functions ..... 305
12.1 Restoring axis configurations and axis couplings ..... 305
12.1.1 Saving a current configuration (\#SAVE CONFIG) ..... 305
12.1.2 Loading or restoring a saved configuration (\#LOAD CONFIG). ..... 308
12.1.3 Clearing a current configuration (\#CLEAR CONFIG) ..... 310
12.2 Axis exchange commands ..... 311
12.2.1 Standard syntax ..... 312
12.2.1.1 Requesting axes (\#CALL AX) ..... 312
12.2.1.2 Releasing axes (\#PUT AX, \#PUT AX ALL) ..... 317
12.2.1.3 Definition of an axis configuration (\#SET AX) ..... 319
12.2.2 Extended syntax ..... 322
12.2.2.1 Requesting axes (\#AX REQUEST) ..... 322
12.2.2.2 Releasing axes (\#AX RELEASE, \#AX RELEASE ALL) ..... 330
12.2.2.3 Definition of an axis configuration (\#AX DEF) ..... 333
12.2.2.4 Load the default axis configuration (\#AX DEF DEFAULT) ..... 337
12.3 Dwell time ..... 339
12.4 Flushing NC channel (\#FLUSH, \#FLUSH CONTINUE, \#FLUSH WAIT) ..... 339
12.5 Cross-block comments (\#COMMENT BEGIN/END) ..... 343
12.6 Waiting for event (\#WAIT FOR) ..... 344
12.7 Adapting minimum radius for tangential feed ((\#TANGFEED)) ..... 345
12.8 Suppressing offsets (\#SUPPRESS OFFSETS) ..... 347
12.9 Settings for measurement ..... 348
12.9.1 Switching measurement type (\#MEAS MODE) ..... 348
12.9.2 Extended programming (\#MEAS, \#MEAS DEFAULT) ..... 349
12.10 Selecting position preset (\#PSET) ..... 353
12.10.1 Deselecting position preset (\#PRESET) ..... 353
12.11 Synchronous operation ..... 355
12.11.1 Programming axis couplings (\#SET AX LINK, \#AX LINK) ..... 355
12.11.2 Extended programming of axis couplings ("SOFT-GANTRY") (\#SET AX LINK, \#AX LINK) ..... 357
12.11.3 Enabling/disabling axis couplings (\#ENABLE AX LINK, \#DISABLE AX LINK) ..... 362
12.11.4 Inquiring coupling state and coupling number via variables ..... 364
12.12 Messages from the NC program ..... 365
12.12.1 Programming a message (\#MSG) ..... 365
12.12.2 Programming message information \#MSG INFO) ..... 368
12.12.3 Including the 'Macro' functionality ..... 369
12.12.4 Writing messages to a file (\#MSG SAVE). ..... 369
12.12.5 Outputting additional informations at block end (\#ADD) ..... 370
12.13 Jerk limiting slope ..... 371
12.13.1 Selecting operating mode (\#SLOPE, \#SLOPE DEFAULT) ..... 372
12.14 Writing and reading SERCOS parameters and commands ..... 374
12.14.1 SERCOS parameters (\#IDENT) ..... 374
12.14.1.1 Non-synchronised write (\#IDENT WR) ..... 374
12.14.1.2 Non-synchronised read (\#IDENT RD) ..... 375
12.14.1.3 Synchronised write (\#IDENT WR SYN) ..... 376
12.14.2 SERCOS commands (COMMAND) ..... 378
12.14.2.1 Non-synchronised write (\#COMMAND WR) ..... 378
12.14.2.2 Synchronised write (\#COMMAND WR SYN) ..... 379
12.14.2.3 Non-synchronised wait (\#COMMAND WAIT) ..... 380
12.14.2.4 Synchronised wait (\#COMMAND WAIT SYN) ..... 381
12.15 Channel synchronisation ..... 383
12.15.1 Sending signals (\# SIGNAL) ..... 385
12.15.2 Removing (broadcast) signals (\#SIGNAL REMOVE) ..... 387
12.15.3 Waiting for signals (\#WAIT) ..... 389
12.15.4 Reading signals without waiting (\#SIGNAL READ) ..... 391
12.15.5 RESET handling ..... 393
12.15.6 Synchronisation scenarios ..... 393
12.16 Rotate the coordinate system in the plane (\#ROTATION ON/OFF) ..... 398
12.17 Automatic axis tracking (C axis tracking) (\#CAXTRACK) ..... 412
12.18 User-defined error output (\#ERROR) ..... 417
12.19 Time measurement (\#TIMER) ..... 420
12.20 Definition of feed axes (\#FGROUP, \#FGROUP ROT, \#FGROUP WAXIS) ..... 422
12.21 Adapt path dynamic limit values (\#VECTOR LIMIT ON/OFF) ..... 425
12.22 Defining a minimum block transition velocity (\#TRANSVELMIN ON/OFF) ..... 429
12.23 Writing machine data (\#MACHINE DATA) ..... 430
12.24 File operations ..... 434
12.24.1 Definition of file names (\#FILE NAME) ..... 434
12.24.2 Renaming a file (\#FILE RENAME) ..... 436
12.24.3 Deleting a file (\#FILE DELETE) ..... 438
12.24.4 Checking existence of a file (\#FILE EXIST) ..... 439
12.24.5 Create and manage backup files ..... 441
12.25 Monitoring the work space and protection space ..... 445
12.25.1 Defining a control area (\#CONTROL AREA BEGIN/END) ..... 446
12.25.2 Selecting/deselecting control areas (\#CONTROL AREA ON/OFF) ..... 449
12.25.3 Clearing control areas (\#CONTROL AREA CLEAR) ..... 450
12.25.4 Monitor additional axes ..... 451
12.25.5 Special features in manual mode ..... 451
12.26 Influence forward/backward motion on path ..... 453
12.26.1 Skipping program sequences (\#OPTIONAL EXECUTION) ..... 453
12.26.2 Clearing backward storage (\#BACKWARD STORAGE CLEAR) ..... 456
12.27 Tool change with active synchronous mode (\#FREE TOOL CHANGE) ..... 457
12.28 Locking program areas for single-step mode (\#SINGLE STEP) ..... 458
12.29 Programmable path override (\#OVERRIDE) ..... 460
12.30 Drive-independent switching of drive functions ..... 461
12.30.1 Synchronised write (\#DRIVE WR SYN) ..... 461
12.30.2 Synchronous waiting for acknowledgement (\#DRIVE WAIT SYN) ..... 462
12.31 Velocity-optimised motion control by segmentation (\#SEGMENTATION) ..... 463
12.32 Enlarging/reducing contours (\#SCALE ON/OFF)) ..... 465
12.33 Punching and nibbling ..... 473
12.33.1 Splitting up motion path and programming (\#STROKE DEF, \#PUNCH ON/OFF, \#NIBBLE ..... 473 ON/OFF)
12.33.2 Further functions ..... 478
12.33.3 Restrictions ..... 479
12.34 Controlling edge machining (\#EDGE MACHINING) ..... 480
12.35 Switching dynamic weighting (\#DYNAMIC WEIGHT) ..... 482
12.36 Weighting of external feedrate (\# FF) ..... 483
12.37 Axis clamping and monitoring ..... 483
12.38 Gantry start-up ..... 485
12.39 Position controller-based axis couplings (\#GEAR LINK) ..... 486
12.40 Settings for turning functions (\# TURN) ..... 489
12.41 Distance to go display in a program section ..... 490
13 Tool geometry compensation (D) ..... 492
13.1 Tool length compensation ..... 493
13.1.1 Axis-specific assignment of tool length compensation (\#TLAX, \#TLAX DEFAULT) ..... 495
13.2 Tool radius compensation (TRC) ..... 498
13.2.1 Direct/indirect deselection of TRC ..... 507
13.2.1.1 Direct selection (G138/G41/G42) ..... 507
13.2.1.2 Indirect selection (G139/G41/G42) with G25 ..... 510
13.2.1.3 Indirect selection (G139/G41/G42) with G26 ..... 513
13.2.2 Direct/indirect deselection of TRC ..... 516
13.2.2.1 Direct deselection (G138/G40) ..... 516
13.2.2.2 Indirect deselection (G139/G40) with G25 ..... 519
13.2.2.3 Indirect deselection (G139/G40) with G26. ..... 522
13.2.3 Perpendicular selection/deselection of TRC (G237) ..... 525
13.2.3.1 Technology functions ..... 527
13.2.3.2 Technology function in single block ..... 531
13.2.4 Selecting inside corner of TRC (G238) ..... 532
13.2.4.1 Restrictions of inside corner selection ..... 534
13.2.5 Direct selection/deselection of TRC without block (G239) ..... 535
13.2.6 Direct selection/deselection of TRC on the path (G236) ..... 540
13.2.6.1 Selecting/deselecting G236 with closed contours ..... 544
13.2.6.1.1 Selection/deselection at inside corners ..... 544
13.2.6.1.2 Selection/deselection at outside corners ..... 545
13.2.7 Generate compensation blocks ..... 546
13.2.8 Reaction on contour change ..... 553
13.2.9 Reaction to change in tool radius ..... 554
13.2.10 Tangential selection/deselection of TRC (G05) ..... 556
13.2.11 Adapting feed of TRC (G10/G11) ..... 559
13.2.12 Selecting/deselecting TRC contour masking (G140/G141) ..... 561
13.2.13 Limits of TRC ..... 562
13.2.14 Programmable additional options of TRC (\#TRC) ..... 564
13.2.14.1 TRC option STRETCH_FACTOR ..... 567
13.2.14.2 TRC option PERPENDICULAR_RADIUS_CHANGE ..... 569
13.2.15 Exception list of commands with active TRC/SRK ..... 572
14 Variables and calculation of variables ..... 575
14.1 Axis-specific variables (V.A.) ..... 578
14.2 Spindle-specific variables (V.SPDL., V.SPDL_PROG.) ..... 582
14.3 Global variables (V.G.) ..... 584
14.3.1 Versioning of NC programs ..... 600
14.4 Self-defined variables (\#VAR, \#ENDVAR, \#DELETE) ..... 602
14.4.1 Global, valid up to end of main program (V.P.) ..... 605
14.4.2 Global, valid throughout main program (V.S.). ..... 607
14.4.3 Local, valid throughout subroutine (V.L.) ..... 609
14.4.4 Cycle variables (V.CYC.) ..... 611
14.4.4.1 Validity and visibility ..... 612
14.4.4.2 Delete variables ..... 614
14.5 External variables (V.E) (\#INIT V.E.) ..... 615
15 Spindle programming ..... 617
15.1 Parameterising spindles ..... 619
15.1.1 Axis parameters ..... 619
15.1.2 Channel parameter ..... 619
15.2 Programming in DIN syntax ..... 624
15.2.1 The spindle M functions ..... 624
15.2.1.1 Move spindle in DIN syntax ((M3/M4/M5) ..... 624
15.2.1.2 Position spindle in DIN syntax ((M19,*. POS) ..... 625
15.2.2 Spindle speed (S) ..... 627
15.2.3 Spindle gear change (M40-M45) ..... 630
15.2.4 Turning functions ..... 633
15.2.4.1 Diameter programming (G51/G52) ..... 633
15.2.4.2 Cutter radius compensation (G40/G41/G42) ..... 635
15.2.4.3 Feedrate per revolution (G95) ..... 637
15.2.4.4 Constant cutting speed (G96/G97/G196) ..... 639
15.2.4.5 Thread cutting with endlessly rotating spindle (G33) ..... 642
15.2.5 Tapping (G63) ..... 646
15.2.6 Tapping (G331/ G332) ..... 648
15.2.7 C axis machining ..... 651
15.2.7.1 Exchange spindles in coordinated motion (\# CAX, \#CAX OFF) ..... 652
15.2.7.2 Face machining (\#FACE, \#FACE OFF) ..... 654
15.2.7.3 Surface machining (\#CYL, \#CYL OFF) ..... 658
15.2.7.4 Switching between face and lateral surface machining ..... 662
15.2.7.5 Tool offsets ..... 663
15.2.8 Gear change (G112) ..... 666
15.2.9 Homing in DIN syntax( G74) ..... 667
15.2.10 Spindle override in DIN syntax (G167) ..... 668
15.3 Programming in spindle-specific syntax ..... 669
15.3.1 The spindle M functions ..... 670
15.3.1.1 Moving spindle in spindle-specific syntax ((M3/M4/M5) ..... 670
15.3.1.2 Positioning spindle in spindle-specific syntax (M19, POS) ..... 672
15.3.2 Spindle speed (REV) ..... 674
15.3.3 User-specific M/H function in spindle-specific syntax ..... 675
15.3.4 Homing in spindle-specific syntax (G74) ..... 675
15.3.5 Spindle override in spindle-specific syntax (G167) ..... 676
15.3.6 Releasing/requesting spindle axes (PUTAX/CALLAX) ..... 677
15.3.7 Adopt tool dynamic data (GET_DYNAMIC_DATA/ DEFAULT_DYNAMIC_D ..... 678
15.3.8 Commanding spindle feedforward control (G135/G136/G137). ..... 679
15.3.9 Spindle feed link (FEED_LINK) ..... 680
15.3.10 Programmable spindle override ..... 683
15.3.11 Acceleration weighting (G130) ..... 683
15.4 Changing the main spindle (\#MAIN SPINDLE) ..... 685
15.5 Synchronous spindle operation ..... 687
15.6 Cross-block synchronisation (Late Sync) ..... 689
15.6.1 Implicit synchronisation ..... 689
15.6.2 Explicit synchronisation (\#EXPL SYN) ..... 690
15.7 Synchronisation of spindle M functions ..... 691
15.8 PLCopen programming ..... 692
15.8.1 MC_Home command ..... 697
15.8.2 MC_MoveAbsolute command ..... 698
15.8.3 MC_MoveAdditive command ..... 699
15.8.4 MC_MoveRelative command ..... 700
15.8.5 MC_MoveSuperImposed command ..... 701
15.8.6 MC_MoveVelocity command ..... 702
15.8.7 MC_Stop command ..... 703
15.8.8 MC_Gearln command ..... 704
15.8.9 MC_GearOut command ..... 706
15.8.10 MC_Phasing command ..... 707
15.8.11 MC_TouchProbe command ..... 708
16 Macroprogramming (\# INIT MACRO TAB) ..... 709
16.1 Nesting macros ..... 710
16.2 Use in mathematical expressions ..... 711
16.3 Separating address letter and mathematical expression ..... 711
16.4 Restrictions ..... 712
17 5-Axis functionality ..... 713
17.1 Rotation Tool Centre Point (RTCP)(\# TRAFO OFF) ..... 713
17.2 Tool Length Compensation (\#TLC ON/OFF) ..... 715
17.3 Orienting tool (\#TOOL ORI CS) ..... 718
17.4 Machine kinematics (\#KIN ID) ..... 720
17.5 Modify kinematic characteristics (\#KIN DATA) ..... 721
17.6 Positioning without compensation motion (\#PTP ON/OFF, \#AX LOCK ALL, \#AX UNLOCK ALL) ..... 722
17.7 Coordinate systems ..... 727
17.7.1 Defining a machining coordinate system (\#CS DEF, \#CS ON/OFF, \#CS MODE ON/OFF). ..... 727
17.7.2 Defining/activating a coordinate system for fixture adaptation (\#ACS). ..... 732
17.7.3 Linkage of coordinate systems ..... 735
17.7.4 Define/activate a basic coordinate system (\#BCS) ..... 739
17.7.5 Effector coordinate system (\#ECS ON/OFF) ..... 741
17.7.6 Temporary transition to the machine coordinate system (\#MCS ON/OFF) ..... 743
17.8 Auxiliary functions for coordinate transformation (\#WCS TO MCS, \#MCS TO WCS) ..... 744
17.9 Auxiliary function to calculate motion limits in the workpiece coordinate system (\#GET WCS ..... 746 POSLIMIT)
17.10 Orientation programming ..... 749
17.10.1 Programming and configuration of 5-axis kinematics (\#ORI MODE) ..... 751
17.10.2 Programming and configuration of 6 -axis kinematics (robot) (\#ORI MODE) ..... 753
17.11 Status \& Turn (IS, IT) ..... 757
18 Programming modulo axes ..... 762
18.1 Positioning on the shortest way ..... 768
19 Extended tool programming ..... 769
19.1 Description of function ..... 769
19.1.1 Tool ID ..... 769
19.1.2 Tool life data recording ..... 769
19.2 Programming commands and variables (V.TOOL.) (\#TOOL DATA, \#TOOL PREP) ..... 770
19.2.1 Weighting factors for tool life and tool life distance ((V.TLM) ..... 771
19.2.2 Reading/removing tool life values (\#TOOL LIFE READ/REMOVE) ..... 772
19.2.3 Refreshing tool data (\#TOOL REFRESH) ..... 773
20 Positioning axes ..... 774
20.1 Independent axes (INDP_SYN, INDP_ASYN) (\#WAIT INDP, \#WAIT INDP ALL) ..... 774
20.2 Oscillating axes (OSC). ..... 778
20.3 Cartesian/kinematic transformation and positioning axes ..... 782
20.3.1 Positioning and shifts ..... 782
20.3.2 Restrictions ..... 782
21 Axis-specific programming ..... 784
21.1 Selecting/deselecting axis compensations in the NC program (COMP) ..... 784
21.2 Distance control (sensed spindles) (DIST_CTRL) ..... 786
21.3 Programmable axis override (OVERRIDE) ..... 790
21.4 Programmable acceleration overload (DYNAMIC) ..... 791
21.5 Synchronising an axis in coordinated motion (SYNC IN / OUT) ..... 792
21.6 Programming an axis polynomial (POLY) ..... 794
21.7 Set an axis position in the channel ..... 797
21.8 Lifting/lowering an axis (LIFT) ..... 797
21.9 Moving to fixed stop ..... 799
22 Appendix ..... 801
22.1 Overview of commands ..... 801
22.1.1 G functions (G..) ..... 801
22.1.2 M functions (M..) ..... 806
22.1.3 Functions reserved according to DIN and ISG extensions ..... 807
22.1.4 Control block statements (\$..) ..... 808
22.1.5 Additional functions (\#..) ..... 809
22.1.6 Additional axis-specific functions ( $\langle X>[$..]) ..... 816
22.1.7 PLC-Open functions (<X>[MC_..]) ..... 816
22.1.8 Variable programming (V.) ..... 816
22.1.9 Miscellaneous functions ..... 816
22.1.10 Migrated NC commands ..... 817
22.2 Revision history ..... 818
23 References ..... 819

## List of figures

Fig. 1: Defining a workpiece coordinate system using NPV and BPV (for legend, see below) ..... 48
Fig. 2: Overview of additional offsets and coordinate systems ..... 51
Fig. 3: Position in rapid traverse with the parameters ..... 53
Fig. 4: Graphic display of linear interpolation (G01) ..... 54
Fig. 5: Description of circle functions G02 and G03 ..... 55
Fig. 6: Examples of circular interpolation ..... 58
Fig. 7: Displaying helical interpolation at constant pitch ..... 59
Fig. 8: Correcting the helix pitch depending on the direction of rotation ..... 60
Fig. 9: Correcting a helix within the range of $180^{\circ}$ after the programmed target point ..... 61
Fig. 10: Correct a helix within the range of $180^{\circ}$ ahead of the programmed target point ..... 62
Fig. 11: Helical interpolation in the XY plane clockwise ..... 63
Fig. 12: Helical interpolation in the XY plane counter-clockwise ..... 65
Fig. 13: Arc in space defined by starting point (1), interim point (2) and target point (3) ..... 67
Fig. 14: Contour line with coordinate in the first main axis ..... 68
Fig. 15: Contour line with coordinate in the second main axis ..... 70
Fig. 16: Validity range of target point. ..... 72
Fig. 17: Contour line with two straight lines (2 angles each with one target coordinate) ..... 73
Fig. 18: Contour line with 2 straight lines, 2 angles, complete target point 2 ..... 75
Fig. 19: Contour line with 2 straight lines, 2 angles, incomplete target point 2 ..... 77
Fig. 20: Validity range of target points with 2 straight lines ..... 79
Fig. 21: Program the measuring function Type 1 ..... 92
Fig. 22: Programmed path with measuring function Type 1 ..... 93
Fig. 23: Program the measuring function Type 2 ..... 94
Fig. 24: Programmed path with measuring function Type 2 ..... 95
Fig. 25: Program the measuring function Type 3 ..... 96
Fig. 26: Programmed path with measuring function Type 3 ..... 97
Fig. 27: Program the measuring function Type 4 ..... 98
Fig. 28: Programmed path with measuring function Type 4 ..... 99
Fig. 29: Measurement offset between probe position and programmed target position ..... 102
Fig. 30: Glue on a veneer strip ..... 104
Fig. 31: Acceleration at block transition in the default state (corresp. to G08) ..... 109
Fig. 32: Deceleration at block transition with G901 and G900 ..... 110
Fig. 33: Deceleration at block transition with G901 and G900 ..... 110
Fig. 34: Combination of G09 with G901 and G900 ..... 111
Fig. 35: Path-related feed interpolation with G193 ..... 113
Fig. 36: Time-related feed interpolation with G293 ..... 113
Fig. 37: Display of plane selection (G17/G18/G19) ..... 114
Fig. 38: Virtual and mirrored (real) coordinates with G21 ..... 115
Fig. 39: Example of mirroring ..... 117
Fig. 40: Mirroring the target point in the motion block. ..... 118
Fig. 41: Changing the contour when mirroring a full circle. ..... 118
Fig. 42: Effects of mirroring functions on the direction of circular rotation in different planes ..... 119
Fig. 43: Mirroring the selected side with active tool radius compensation ..... 121
Fig. 44: Mirroring a reference point offset G92 ..... 121
Fig. 45: Examples of polynomial contouring ..... 127
Fig. 46: Changing the removal volume Vz over time at an inside corner of $90^{\circ}$ and at constant feedrate ..... 129
Fig. 47: Representation of feed at a circular inside contour ..... 130
Fig. 48: Relationship between centre point offset Dm and the calculated radius "radius" ..... 144
Fig. 49: Area of permissible programmed centre points ..... 145
Fig. 50: Circle centre point shift in the case of G165 ..... 147
Fig. 51: Example of ramp time weighting with $\mathrm{G} 132 / \mathrm{G} 133 / \mathrm{G} 233$ ..... 155
Fig. 52: Ramp time weighting with G134 and with circular interpolation ..... 156
Fig. 53: Insert a chamfer between two straight lines ..... 164
Fig. 54: Insert a chamfer between two arcs ..... 164
Fig. 55: Error due to direction reversal ..... 165
Fig. 56: Inserting an arc between two straight lines ..... 166
Fig. 57: Inserting an arc between two circles (angle $\alpha \geq 180^{\circ}$ ) ..... 167
Fig. 58: Inserting an arc between two circles (angle $\alpha<180^{\circ}$ ) ..... 167
Fig. 59: Manual mode and its options ..... 168
Fig. 60: Significance of rotation parameters in the main plane (example G17): ..... 190
Fig. 61: Program feedrate using F word ..... 201
Fig. 62: Program feedrate using F and E words ..... 201
Fig. 63: Effect of E word on inserted contour elements (here G261) ..... 202
Fig. 64: Application example of parameter calculation ..... 224
Fig. 65: Illustration of the effect of indirect $P$ parameters ..... 228
Fig. 66: Permitted and impermissible jumps in the \$GOTO command ..... 238
Fig. 67: Trim a contour ..... 248
Fig. 68: Line-by-line surface machining ..... 249
Fig. 69: Definition of corner distance ..... 259
Fig. 70: Definition of corner deviation ..... 259
Fig. 71: Contour from programming G61 - G261/G260 ..... 260
Fig. 72: Example of skipping a short block N05 when contouring ..... 262
Fig. 73: Some single blocks (N20, N30 and N40) are too short but the target point is outside the minimum block length. ..... 263
Fig. 74: Single blocks (N20, N30 and N40) are too short but the sum of all blocks exceeds the minimum sys- tem-specific block length. ..... 263
Fig. 75: Multiple blocks (N10, N20 and N30) are too short but the sum of all blocks exceeds the minimum system-specific block length. ..... 264
Fig. 76: Some single blocks (N20, N30 and N40) are too short but contouring is deselected as of block N20.. ..... 264
Fig. 77: Synchronisation without contour-relevant actions during contouring ..... 265
Fig. 78: Synchronisation without contour-relevant actions after contouring ..... 265
Fig. 79: Characteristic in the transition section ..... 267
Fig. 80: Contouring with corner deviation ..... 271
Fig. 81: Corner distance contouring ..... 273
Fig. 82: Maximum corner distance of block N20 independent of the block lengths of N10 and N20 (DIST_WEIGHT = 0\%) ..... 275
Fig. 83: Maximum corner distance of block N20 subdivided relative to the block lengths of N10 and N30 (DIST_WEIGHT = 100\%) ..... 275
Fig. 84: Contour with interim point ..... 279
Fig. 85: Dyn. optimised contouring of the entire contour specifying corner deviation ..... 283
Fig. 86: Block limit before contouring curve ..... 284
Fig. 87: Block limit within contouring curve ..... 285
Fig. 88: Block limit after contouring curve. ..... 286
Fig. 89: Block limit after contouring curve. ..... 287
Fig. 90: Examples of combining transition types 1 and 2 ..... 293
Fig. 91: Contour in the programming example (no. refers to 1st programming example) ..... 296
Fig. 92: Contour resulting from programming example ..... 299
Fig. 93: Insert transition polynomials ..... 301
Fig. 94: Generating spline curves for HSC programming ..... 302
Fig. 95: Mode of operation of \#FLUSH between 2 motion blocks ..... 340
Fig. 96: Mode of operation of \#FLUSH CONTINUE between 2 motion blocks ..... 341
Fig. 97: Mode of operation of \#FLUSH WAIT between 2 motion blocks ..... 342
Fig. 98: Programming the tangential feedrate. ..... 345
Fig. 99: Positions of the $X$ axis in machine coordinates / programmed coordinates ..... 354
Fig. 100: Mechanical gantry operation ..... 357
Fig. 101: Programmable gantry operation ("soft" gantry) ..... 357
Fig. 102: Acceleration on the programmed path ..... 371
Fig. 103: Parameters of the acceleration profile ..... 371
Fig. 104: Velocity curve depending on the programmed path. ..... 373
Fig. 105: Example application: double-column machine with tool changer ..... 383
Fig. 106: Sequence in case of shared access to a resource ..... 384
Fig. 107: Synchronisation of 2 decoders on 2 channels ..... 393
Fig. 108: Synchronisation between decoder and interpolators on 3 channels ..... 394
Fig. 109: Synchronisation between decoder and interpolator of one channel ..... 396
Fig. 110: Significance of rotation parameters in the main plane (example G17): ..... 398
Fig. 111: Tracking the rotary $C$ axis tangentially relative to the $x-y$ contour ..... 412
Fig. 112: Diagram of backup function ..... 442
Fig. 113: Creating backup files ..... 443
Fig. 114: Definition of 3D control areas (cylindrical, polygonal). ..... 445
Fig. 115: Example of cylindrical workspace areas in an application ..... 445
Fig. 116: Splitting up linear blocks ..... 473
Fig. 117: Splitting up circular blocks ..... 474
Fig. 118: View of the difference between source and target axis ..... 486
Fig. 119: Distance to go display in a program section ..... 491
Fig. 120: Compensate tool length by compensating motion ..... 493
Fig. 121: Example of tool length compensation.... ..... 494
Fig. 122: Assignment rule of tool length compensation ..... 495
Fig. 123: Contour approach motions in G17, G18, G19 with tool length compensation in constant orientation ..... 497
Fig. 124: Mode of operation and terms of tool radius compensation ..... 498
Fig. 125: Contour example with G237 ..... 526
Fig. 126: Contour example with technology function 1 ..... 528
Fig. 127: Contour example with technology function 2 ..... 529
Fig. 128: Contour example with technology function 3 ..... 530
Fig. 129: Contour example with technology function in single block ..... 531
Fig. 130: Contour example for inside corner selection (G238) ..... 533
Fig. 131: Motion block sequence of G239_a position in both main axes ..... 535
Fig. 132: Motion block sequence of G239_only one main axis programmed ..... 536
Fig. 133: Motion block sequence of G239_only one main axis programmed ..... 537
Fig. 134: Motion block sequence of G239_Program the 2nd main axis in 2nd motion block ..... 537
Fig. 135: Motion block sequence of G239_Program the 2nd main axis as for the 1st selection ..... 538
Fig. 136: Motion block sequence of G239_Centre point does not match starting and end points ..... 539
Fig. 137: Selection and deselection of closed contours at inside corner ..... 544
Fig. 138: Selection and deselection of closed contours at outside corner ..... 545
Fig. 139: Example of contour transition on straight lines for linear-linear block sequence ..... 547
Fig. 140: Example of contour transition to an arc for linear-linear block sequence ..... 547
Fig. 141: Example of a selection change without deselection ..... 553
Fig. 142: Tool radius change within linear block ..... 554
Fig. 143: Tool radius change within circular block ..... 555
Fig. 144: Selection and deselection of TRC in tangential mode ..... 557
Fig. 145: Selection and deselection of TRC in tangential mode ..... 558
Fig. 146: Adapting feedrate with compensated circular interpolation ..... 559
Fig. 147: Masking of N20 to avoid contour violation. ..... 561
Fig. 148: Example of detected contour violation ..... 562
Fig. 149: Example of undetected contour violation. ..... 563
Fig. 150: Illustration of kerf masking ..... 565
Fig. 151: Validity of self-defined V.CYC. variables ..... 612
Fig. 152: Validity of V.CYC. variables of the same name ..... 613
Fig. 153: Correct use of DIN syntax and spindle-specific syntax ..... 617
Fig. 154: Reference points and diameter programming for turning ..... 633
Fig. 155: Orientation of cutter edge to machining plane ..... 635
Fig. 156: Tool gauging for tool offset compensation ..... 636
Fig. 157: Spindle speed with active G96 ..... 639
Fig. 158: Value of thread pitch for longitudinal thread ..... 643
Fig. 159: Value of thread pitch for tapered thread ..... 643
Fig. 160: Illustration of geometry example ..... 644
Fig. 161: Face machining ..... 654
Fig. 162: Front view of face machining process ..... 654
Fig. 163: Main places of face machining ..... 655
Fig. 164: Lateral surface machining ..... 658
Fig. 165: Tool offsets for face machining ..... 663
Fig. 166: Tool offsets for lateral surface machining ..... 665
Fig. 167: Diagram of synchronisation of the spindle $M$ function ..... 691
Fig. 168: Motion control with/without RTCP ..... 713
Fig. 169: Motion control with RTCP ..... 715
Fig. 170: When the tool length is changed, TLC transforms $\Delta L$ in each cycle. ..... 716
Fig. 171: Tool aligned perpendicularly to the $\mathrm{X}-\mathrm{Y}$ machining plane ..... 718
Fig. 172: Motion control with/without \#PTP ..... 724
Fig. 173: Machining on an inclined plane ..... 728
Fig. 174: Definition of a CS by 3 rotations referred to the new axes ..... 729
Fig. 175: Definition of a CS by 3 rotations about fixed axes in space ..... 729
Fig. 176: The combination of ACS and CS permits machining on an inclined plane with a slanted clamped workpiece. ..... 735
Fig. 177: Activating or changing the ACS without deselecting the CS's which are already active ..... 736
Fig. 178: Result of a CS linkage depending on the sequence of selection (CS[1] - CS[2] or CS[2] - CS[1]). ..... 737
Fig. 179: Linkage of coordinate systems ..... 738
Fig. 180: Linkage with basic coordinate system \#BCS ..... 740
Fig. 181: Machining in a slanting hole ..... 741
Fig. 182: Example of motion limits in the ZX plane in the current WCS ..... 748
Fig. 183: Orientation vector at 5 -axis head ..... 749
Fig. 184: Orientation vector on robot ..... 750
Fig. 185: The intersection of the hand axes (arrowhead) is in the (blue) base area. ..... 758
Fig. 186: Status bit 1 for robots with an offset between axis A3 and axis A5 ..... 759
Fig. 187: Status bit 2 for axis angle position $A 4=0^{\circ}$ and $A 4=180^{\circ}$. ..... 759
Fig. 188: Motion diagram of path axis compound/independent axes ..... 774
Fig. 189: Grinding with an oscillating axis ..... 778
Fig. 190: Positioning procedure with pendulum movement ..... 780
Fig. 191: Synchronised cutting ..... 792
Fig. 192: Single-row lifting ..... 799

## 1 <br> Brief description

The controller processes syntax according to DIN 66025 or customary design and syntax elements according to extensions:

- Text-based program names
- Comprehensive parameter calculation (local and global parameters)
- Access to internal control data such as positions, measurement and tool data, zero offsets etc., by plain text designations (V.A.name, V.G.name)
- Definition of plain texts to designate free parameters in the NC program (V.L.name, V.S.name, V.P.name)
- Control block statements based on the "C" programming language, for example:
- Conditional jumps: \$IF, \$ELSEIF, \$ELSE, \$ENDIF, \$SWITCH, \$CASE, \$DEFAULT,\$ENDSWITCH, \$BREAK
- Counting loops: \$FOR, \$ENDFOR, \$CONTINUE, \$BREAK
- Loops with running condition: \$WHILE, \$ENDWHILE, \$CONTINUE, \$BREAK
- Loops without running condition: \$DO, \$ENDDO, \$CONTINUE, \$BREAK
- Jumps within the same NC program level: \$GOTO
- Distinction between global (accessible by all main programs) and local subroutines (accessible only from the associated main program)
- Mathematical expressions, e.g.:
- Basic standard arithmetic operations: + , - , *, / , ** , MOD
- Numerical functions such as ABS,SQR,SQRT,EXP,LN,DEXP,LOG
- Trigonometric functions such as SIN,COS,TAN,ASIN,ACOS,ATAN
- Conversion functions such as INT,FRACT,ROUND
- Technology commands with configurable effect of each function:
- Additional functions (MO....M65535)
- Auxiliary functions (H0....H65535)
- Spindle functions (S, M3, M4, M5, M19)
- Tool functions (T, D)
- Processing of coordinate notation $\mathrm{A}, \mathrm{B}, \mathrm{C}, \ldots, \mathrm{U}, \mathrm{V}, \mathrm{W}$, if they are not used in any other way, and/or programming of strings (e.g. X_ACHSE, SPINDEL_1 ...).


## Notice

The Appendix contains a complete list of NC commands [ [ 801].

## Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

## 2 Basic principles of programming

### 2.1 Typographical representation of syntax

The convention below applies to the typographical representation of NC command syntax described in this documentation:

| Bold | Obligatory syntax elements of NC command. |
| :--- | :--- |
| $[$ ] | Optional non-recurrent occurrence |
| $\}$ | 0 or multiple occurrence |
| $\mid$ | Alternative between several symbols ("or") |
| $<>$ | Mathematical expression [> 30] or string |

### 2.2 Character and number formats

### 2.2.1 Character set and file format

The decoder used processes NC control data in ASCII format. NC control data can be generated using a programming system integrated in the operating system or in an editor.

## The CNC processes the following characters:

$\left.\begin{array}{llr}\text { Characters }= & \begin{array}{l}\text { letters } \\ \text { digits } \\ \text { special characters } \\ \text { control characters }\end{array} & \begin{array}{l}\text { and/or } \\ \text { and/or } \\ \text { and/or }\end{array} \\ & \\ \text { letters } & \{\text { A B C } \ldots \text { Z ab c } \ldots \text { z }\}\end{array}\right]$

## Restrictions with NC filenames

The following characters are not permitted for filenames:
special characters
control characters
("] \# \$ ;
TAB, CARRIAGE RETURN, LINE FEED

### 2.2.2 Numerical input

The input field for numerical values is limited by the internal numeric display.
This numeric display allows motion paths in the range of 200 m at a resolution of $0.1 \mu \mathrm{~m}$.
By changing the internal conversion factors in a specification file, the input of positions, feeds, etc., may also be used in different units than mm or inch.

## Example

## Numerical values as integers or decimal point inputs:

- Decimals are generally separated by a "." whereby leading zeroes may be omitted.
- Depending on the configuration and programming, inputs for lengths and position values are given in millimetres or inches; angles are entered in degrees [ ${ }^{\circ}$ ] or gons [gon].


## Input format:

| Values: $[\mu \mathrm{m}]$ | Inputs: $[\mathrm{mm}]$ |
| :--- | :--- |
| 0,1 | 0.0001 or .0001 |
| 1 | 0.001 or .001 |
| 10 | 0.01 or .01 |
| 100 | 0.1 or .1 |
| 1000 | 1.0 or .1 |
| 10000 | 10.0 or .10 |
| 100000 | 100.0 or .100 |
| 1000000 | 1000.0 or .1000 |

## Example

## Numerical inputs as hexadecimal numbers:

- These numbers are enclosed in '...' and start with the character combination 16\#, $0 x$ or alternatively H . Following leading zeros or blanks are ignored.
- The characters $A$ to $F$ or a to $f$ can be used for an additional six numerals.

Format:'16\#<A...F, a..f, 0..9>' oder '0x<A...F, a..f, 0..9>'or 'H<A...F, a..f, 0..9>'
Input format:

| '16\#FA1B' | is equivalent to decimal value 64027 |
| :--- | :--- |
| '0x0ED2' | is equivalent to decimal value 3794 |
| 'H1869f' | is equivalent to decimal value 99999 |

## Example

## Numerical inputs as binary numbers (dual numbers):

- These numbers are enclosed in '...' and start with the character combination 2\#, 02\# or alternatively B. Following leading zeros or blanks are ignored.
- The two digits are displayed as zero (0) and one (1).

Format: '2\#<0..1>' or '02\#<0..1>' or 'B<0..1>'
Input format:

| '2\#1010011' | is equivalent to decimal value 83 |
| :--- | :--- |
| '02\#010011' | is equivalent to decimal value 19 |
| 'B11101010' | is equivalent to decimal value 234 |

## Example

Numerical inputs as octal numbers:

- These numbers are enclosed in '...' and start with the character combination 8\# or 08\#. Following leading zeros or blanks are ignored.
- The digits 0 to 7 are used for representation.

Format: '8\#<0..7>' or '08\#<0..7>'
Input format:
'8\#12345'
is equivalent to decimal value 5349
'08\#0107302'
is equivalent to decimal value 36546

### 2.3 Structure of NC control data: NC programs

Definitions for understanding:
NC programs are part of the NC control data in addition to tool data, zero offset data etc. NC programs describe the flow of machining processes.
Separating characters indicate the end of a sequence of characters (character string). The decoder evaluates the following separating characters:

| CR | LF | ETX | TAB | 10 | Blank | $($ | $;$ | $"$ | $]$ | , | $\#$ | $\$$ |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

## Notice

These characters may not be used in file names or as part of \% program names by the NC main program or subroutines (local, global); otherwise an error is output.

Comments contain non-decodable ASCII information which must be placed between the characters "(" and ")" or after the character ";". If an additional "(" is found within a comment, a matching ")" is expected. The block end character and the file end character end the comments.

Mathematical expressions [ $>30$ ] are composed of numerals, parameters, operators, functions, etc. They are evaluated by an integrated calculation function.
Examples: "100", "100+20", "100+P10", "100+PP10", "100*[2+P3]", "DEXP[2]", "100+SIN[P10+PP20]".

## An NC program consists of:

- Comments
- Definitions of local subroutines, consisting of:
- String "\%L" followed by a subroutine name
- A number of NC program blocks
- A code identifying subroutine end (M17 or M29)
- Definition of the main program, consisting of:
- String "\%" followed by a main program name
- A number of NC blocks
- A code identifying main program end (M02 or M30).


## Notice

If a file outside the comments contains a first character that is neither a separating character nor a "\%", the character is evaluated as the first character of an unnamed main program. It also means that no block numbers may be programmed in front of "\%".

### 2.4 NC block structure

An NC block consists of a

- block number (optional)
- a number of words (NC commands)
- block end identifier

An NC block has a maximum length of 4000 characters.
The use of \# commands (see Section Special functions [ 305]) excludes the programming of other words in the NC block (except for the block number).

As a rule each block begins with a block number consisting of an N character followed by a mathematical expression. This expression is mapped in the display data and rounded off as an integer.
The use of block sequences (SEQUENCE) and program jumps (\$GOTO) requires the unambiguous and ascending programming of block numbers.

Otherwise, the block number is of no significance for program flow. In this case the block number does not even need to be programmed in ascending order.

## Example of a block structure:



NC commands conforming to DIN 66025 need not be compulsorily separated by spaces or tabs. When programming text commands deviating from DIN (control block statements, special functions, etc.), the syntax requires separating characters which are also useful to structure an NC program.

## Examples of an NC program structure:

| Without | Partial | Complete |
| :---: | :---: | :---: |
| numbering | numbering | numbering |
| \% 100 | \% 100 | \% 100 |
| "Block 1" | N10 "Block 1" | N10 "Block 1" |
| "Block 1" | "Block 2" | N20 "Satz 2" |
| "Block 1" | N20 "Satz 3" | N30 "Satz 3" |
| - | "Block 4" | N40 "Satz 4" |
| - | - | - |
| - | - |  |
| M30 | M30 | N700 M30 |

Words must be distinguished according to their significance into:

- Geometrical information (e.g. positions),
- Technological information (e.g. spindle speed, feedrate, clockwise spindle rotation),
- Information on program flow control (called control blocks such as counting loops),
- Arithmetic information (e.g. calculation of a variable, parameter calculation).

Several words may be in a block (exception: special commands from Section Special commands [ $>305$ ]) whereby the processing sequence of control data within the block is defined by the controller. The programmer can then enter the individual words of an NC block in any order without this having any effect on processing. This programming manual contains special notes to point out exceptions.
The block end identifier normally consists of a combination of the control characters "CR" and "LF".

### 2.4.1 Skipping NC blocks '/'

Skipped blocks permit optional processing stages such as measuring loops, test blocks or dummy stages within an NC program.

### 2.4.1.1 Standard skipping

Specific NC blocks can be skipped by prefixing them with a "/" character. The controller ignores NC blocks if the function "Skip block" is activated before main program start by a BOOL command on the operating console (HMI) or by the PLC.
/ N3412 X100 ..

In Builds up to V3.01.3020.01, any change in skip settings while an NC program is active only becomes effective at the next main program start. The extended skipping [> 26] function is then available in higher Builds.

## / N..

### 2.4.1.2 Extended skipping (skip levels)

## Release Note

This function is available as of Build V3.01.3021.00.

By combining a slash "/" and a numeral, you can set up to 10 different skip levels in the NC program (e.g. /5 N3412...). The skip levels are activated on the operating console (HMI) or by the PLC by a 32 bit command before main program start. Multiple skip levels can be activated at the same time.

In the Extended Skipping function, changes in skipping settings take effect immediately while the NC program is active. Defined breakpoints can be implemented, e.g. by M functions followed by \#FLUSH WAIT, to ensure that these skipping setting changes are safely accepted and become effective in the NC program.
For more information see [FCT-M6].
For compatibility reasons the skip level without a number "/" and "/1" have the same meaning.

```
l<1-10>N.
```


## Programing Example

## Extended skipping

```
%skip_levels
NO5 GO XO YO Z0
/1 N10 G74 X1 Y1 Z1
N20 G01 F1000 X10 Y10
/2 N30 Z3
```

```
N40 X-1 Y-2
/10 N90 #CS ON [0,0,0,0,0,45]
N95 X30 Z50
/ N99 G92 X0
N999 M30
```


### 2.4.2 Block-specific comments

Explanatory comments can be inserted at almost any position in an NC program, even at the start of a program. Comments have no effect on the processing of an NC program.

Comments start with a "(". If the comment is extended up to block end, it is sufficient to separate it by "(". For comments within an NC block the corresponding ")" character is also required.
Alternatively, a comment can start with a semicolon ";". This comment type always extends up to block end.

Comments may be of any length. However, the maximum number of characters per block may not be exceeded.

It is possible to nest comments.

Programing Example

## Comments in the NC code

\% 100 (comment in full brackets)

```
N200 ... (comment only with open bracket
N300 (comment (nested comment))
N500 X10 (comment in block) Y20
N700 ... ; Comment after semicolon
```

N999 M30

## Notice

Comments on any number of program lines (see Section Cross-block valid comments [ 343]) may be made using the special commands \#COMMENT BEGIN and \#COMMENT END.

### 2.4.3 Line break in NC block 'l'

With some NC commands the syntax sequence of an NC block may be separated into two or several lines by inserting line breaks ' $\mid$ ' to improve clarity and readability. The line break character ' $\$ ' may only be placed after complete syntax terms.

Separated blocks may only be used to continue an associated syntax sequence.

## Notice

## EXCEPTION:

At the beginning of a separated block, a new block numeral " $\mathrm{N} . .$. " is permissible.

## In general the line break 'l' can be used in brackets for:

- Axis-specific programming X[...]
- Spindle-specific programming S[...]
- PLCopen commands S[MC_...]
- Cycle programming L CYCLE [...]
- Declaration of self-defined variable arrays

And in general for:

- DIN/ISO programming G01 X10 ...


## Attention

NC commands from the group of additional functions (\# commands) may not be split by a line break. A separate note is provided for exceptions!

## Programing Example

Line break in NC block 'l'

```
N10 X[INDP_ASYN G90 POS=100 G01 FEED=2500
N20 SLOPE_TYPE=STEP M23 M56 H78]
N...
N10 S[M4 REV5000 M11] S2[M3 REV5000 M34] S2[REV1000 M3 POS=45 M19 \
N20 M11 M12 H56]
N...
N10 S[MC_MoveAbsolute Position=133 Velocity=1000 Acceleration=500
N20 Deceleration=600 Jerk=200 Direction=2]
N...
N... L CYCLE [NAME=pocketmill.cyc @P1=100 @P2=80 @P3=5 @P5=15 \
@P6=80 @P7=60 @P8=10 @P9=65 @P10=50 @P11=3 @P12=1 \
@P13=2 @P14=1 @P15=-1 @P16=40 @P17=3]
N10 G60 G90 F1000 G01 \
N20 X100 Y150 Z250
N...
#VAR
V.P.ARRAY_1[3][6] = [10,11,12,13,14,15, \
    20,21,22,23,24,25, \
    30,31,32,33,34,35]
#ENDVAR
```


### 2.5 Word structure

A word consists of address letters, arithmetic expressions and texts. <string>. The meaning of the individual address letters is described in the following sections. An overview is provided in Section "Assigned address characters [> 41]".

### 2.5.1 Mathematical expressions

Consisting of:

- Numerical values broken down into:
- Integers <int> and
- Decimal numbers <double>
- Arithmetic expressions <expr> are composed of:
- numerical values
- operators
- functions
- parameters
- Variablen
- macros

Example: [[sin["MAX_ANZ" * 30.00] + P2] / V.G.BLOCK_NR]

### 2.5.1.1 Integers <int>

Integers are specified without a decimal point. The internally permissible value range corresponds to "long integer" variables in the C programming language, i.e. the range from $-2.14^{*} 10^{9}$ to $+2.14^{*} 10^{9}$.
If internal calculation uses the unit $0.1 \mu \mathrm{~m}$, the numerical range when values are entered in mm is between $-2.14^{*} 10^{5}$ and $+2.14^{*} 10^{5}$. This corresponds to a motion path of over 400 metres. Due to an internal limitation in the position controller, the motion path is limited to half, i.e. to just over 200 metres.
Where negative numerals are preceded by "-", positive numerals do not require the prefix "+".

### 2.5.1.2 Decimal numbers <double>

Decimal values are specified with a decimal point or alternatively, in exponential notation, with a lowercase ' e ' (also referred to as scientific notation). The value range corresponds to integer numerals. If internal calculation uses the unit $0.1 \mu \mathrm{~m}$, input values in mm are then provided with 4 digits after the decimal point.

Where negative numerals are preceded by "-", positive numerals do not require the prefix "+".

## Examples of decimal numbers:

| 123.3456 | 0.6789 | .6789 | -345.56 | +78.987 |
| :--- | :--- | :--- | :--- | :--- |
| $12 e 5$ | +2.5 e 6 | -4 e 7 | $-523.6 \mathrm{e}-3$ |  |

### 2.5.1.3 Arithmetic expressions <expr>

The usual calculation rules apply to handling arithmetic expressions:

- Order-of-operations rule
- the parenthesis rule, whereby square brackets "[ ]" must be used. Round parenthesis "(...)" are for comments.

Parameters are often used in arithmetic expressions. The notation of parameters is:

- P followed by an integer, e.g. P12.


## Example of an arithmetic expression:

P5 = [[sin[R1*30.00] + P2] / P5]

Macros (strings) may be assigned to arithmetic expressions and parts of them.
A macro name leads to a macro content which is analysed. Recursive handling is also possible.
Macro names must be placed in quotation marks. When decoded, the notation is case-sensitive (uppercase/lowercase).

Nested strings are identified by a preceding 'l' before the double quotation marks. Make sure that complete nesting levels are always grouped in a macro, i.e. adding ' $[$ ' at the start and ']' at the end of macro content should have no effect on the result of the mathematical expression.

## Programing Example

 Nested macros
## Correct:

```
N10 "STRING1" = "COS[\"STRING2\"]"
N2O "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"
N40 X[-2 * "STRING1" + "STRING2" + "STRING3"] (Move to X60)
```

M30

## Wrong:

```
Only complete nesting levels may be compiled
```

in the string.
N10 "STRING1" = "COS["
N20 "STRING2" = "90]"
N30 "STRING3" = " \"STRING1\" \"STRING2\" "

Macros defined in the NC program are valid program global.
Section "Macroprogramming (\# INIT MACRO TAB) [ 709]" describes how to program macros outside mathematical expressions.

## Overview of all available calculation operations:

Basic types of calculation:

| Addition | + | $\mathrm{P} 1=\mathrm{P} 2+\mathrm{P} 3+0.357$ |
| :--- | :---: | :--- |
| Subtraction | - | $\mathrm{P} 1=\mathrm{P} 2-0.031$ |
| Multiplication | $/$ | $\mathrm{P} 1=\mathrm{P} 2 *[\mathrm{P} 3+0.5]$ |
| Division | $* *$ | $\mathrm{P} 1=\mathrm{P} 2 * \mathrm{P} 3 /[\mathrm{P} 5+\mathrm{P} 6]$ |
| Exponential calculation | MOD | $\mathrm{P} 1=11$ MOD $3 \quad(->2)$ |
| Modulo calculation | 2 to the power P3 $)$ |  |

## Numerical functions:

| Absolute value formation | ABS [..] | $\mathrm{P} 1=\mathrm{ABS}$ [P2 - P4] |
| :---: | :---: | :---: |
| Squaring | SQR [..] | $\mathrm{P} 1=\mathrm{SQR} \mathrm{[P2]} \mathrm{+} \mathrm{SQR} \mathrm{[P3]}$ |
| Square root | SQRT [..] | $\mathrm{P} 1=\mathrm{SQRT}$ [SQR[P2]+SQR[P3]] |
| e function | EXP [..] | $\mathrm{P} 1=\operatorname{EXP}[\mathrm{P} 2 \times \mathrm{P} 4]$ |
| Natural logarithm | LN [..] | $\mathrm{P} 1=\mathrm{LN}[\mathrm{P} 2]+\mathrm{LN}$ [P3] |
| To the power of ten | DEXP [..] | $\mathrm{P} 1=\mathrm{DEXP}$ [P2] |
| Common logarithm | LOG [..] | $\mathrm{P} 1=\mathrm{LOG}$ [P2] |

## Attention

In the case of LN, LOG and SQRT the argument must always be greater than 0 !

## Bit operators:

| AND operation | $\&$ | $\mathrm{P} 1=\mathrm{P} 2 \& \mathrm{P} 3$ |
| :--- | :---: | :--- |
| OR operation | I | $\mathrm{P} 1=\mathrm{P} 2 \quad \mathrm{P} 3$ |
| Exclusive OR | $\wedge$ | $\mathrm{P} 1=\mathrm{P} 2 \wedge \mathrm{P} 3$ |
| Complement | $\mathrm{INV}[.]$. | $\mathrm{P} 1=\mathrm{INV}[\mathrm{P} 2]$ |

## Attention

The operands may be any positive mathematical expression or number within the range 0 ... $2^{\wedge} 32-1$ (UNS32). Negative expressions or numerals are not allowed. Floating point numbers are converted into integers.

The result of a bit operation is always within the range of $0 . . .2^{\wedge} 32-1$ (UNS32).

Logic operators:

| AND operation | \& / AND | $\begin{aligned} & \mathrm{P} 1=\mathrm{P} 2 \& \& \mathrm{P} 3 \\ & \mathrm{P} 1=\mathrm{P} 2 \text { AND } \mathrm{P} 3 \end{aligned}$ |
| :---: | :---: | :---: |
| OR operation | \|| / OR | $\begin{aligned} & \mathrm{P} 1=\mathrm{P} 2 \text { \|\| } \mathrm{P} 3 \\ & \mathrm{P} 1=\mathrm{P} 2 \text { OR P3 } \end{aligned}$ |
| Exclusive OR operation | XOR | P 1 = P2 XOR P3 |
| NOT operation | NOT[..] | $\begin{aligned} & \mathrm{P} 1=\mathrm{NOT}[\mathrm{P} 2] \\ & \mathrm{P} 1=\mathrm{NOT}[1] \quad(\mathrm{P} 1=0) \\ & \mathrm{P} 1=\operatorname{NOT}[0.5] \quad(\mathrm{P} 1=0) \\ & \mathrm{P} 1=\operatorname{NOT}[0.49](\mathrm{P} 1=1) \\ & \mathrm{P} 1=\operatorname{NOT}[0] \quad(\mathrm{P} 1=1) \end{aligned}$ |

## Attention

Operands may be any positive mathematical expression or numeral. Negative expressions or numerals are not allowed.

A floating point numeral is evaluated as TRUE (1) if its value is $\boldsymbol{>}$ or $\mathbf{= 0 . 5}$.

## Comparison operators:

Loops (Section Statements for influencing NC program flow [ 230]) require comparison expressions. Verification can be conducted as follows:

| Equality | == | \$IF P1 == 5 |
| :---: | :---: | :---: |
| Inequality | != | \$IF P1 ! = 5 |
| Greater than or equal to | >= | \$IF P1 >= 10 |
| Less than or equal to | <= | \$IF P1 <= 10 |
| Less than | $<$ | \$IF P1 < 10 |
| Greater than | > | \$IF P1 > 10 |

## Operator priorities:

The priorities of available operators are listed in descending order. 10 is the highest and 1 is the lowest priority.

| Priority | Operator | Description |
| :--- | :---: | :--- |
| 10 | $* *$ | Power |
| 9 | $*, /$ | Multiplication, division |
| 8 | ,+- | Addition, subtraction |
| 7 | $\&$ | Bitwise AND |
| 6 | $\wedge$ | Bitwise exclusive OR |
| 5 | $<=,>=,==,<,>,!=$ | Comparison operators |
| 4 | $\& \&$, AND | Logic AND |
| 3 | XOR | Logic exclusive OR |
| 2 | $\\|$, OR | Logic OR |
| 1 |  |  |

Possible truth values are:

| True | TRUE | \$IF V.A.MERF.X == TRUE |
| :--- | :---: | :--- |
| Not true | FALSE | \$WHILE V.G.WZ[2].OK == FALSE |

## Attention

Processing truth values:
For TRUE, the value 1 is used in the controller.
For FALSE, the value 0 is used in the controller.

Trigonometric functions (angles specified in degrees):

| Sine | SIN [..] | P1 = SIN [P2] |
| :---: | :---: | :---: |
| Cosine | Cos [..] | $\mathrm{P} 1=\operatorname{CoS~[P2]~}$ |
| Tangent | TAN [..] | P1 = TAN [P2] |
| Cotangent | COT [..] | P 1 COT [P2] |
| Arcsine | ASIN [..] | P1 = ASIN [P2] |
| Arccosine | ACOS [..] | $\mathrm{P} 1=\mathrm{ACOS}[\mathrm{P} 2]$ |
| Arctangent | ATAN [..] | P1 = ATAN [P2] |
| Arctangent with 2 arguments | ATAN2 $[y, x]$ | $\begin{aligned} & \text { P1 = ATAN2 }[100,100] \\ & \left(->\text { result is } 45^{\circ}\right) \end{aligned}$ |
| Arc cotangent | ACOT [..] | $\mathrm{P} 1=\mathrm{ACOT}[\mathrm{P} 2]$ |

## Attention

With the numerical functions ASIN and ACOS, the argument must always be between -1 and +1 . For the numerical function TAN, the argument should not assume the values -90, 90, $270 \ldots$ degrees.
For the numerical function COT, the argument should not assume the values... $-180,0,180 \ldots$ degrees.

The numerical function ATAN2 results in $x!=0$ for the angle of a position relative to the $X$ axis in the correct quadrant.

Special case: For ATAN2[0.0] ( $\mathrm{x}=0$ and $\mathrm{y}=0$ ), the result is always 0 .

Transformation functions:

| Remove places after decimal point | INT [..] | $\mathrm{P} 1=\operatorname{INT}[123.567] \quad(\mathrm{P} 1=123)$ |
| :---: | :---: | :---: |
| Remove integer part | FRACT [..] | $\begin{aligned} & \mathrm{P} 1=\mathrm{FRACT}[123.567] \\ & (\mathrm{P} 1=0.567) \end{aligned}$ |
| Round up to integer | ROUND [..] | $\begin{aligned} & \mathrm{P} 1=\operatorname{ROUND}[77.5] \quad(\mathrm{P} 1=78) \\ & \mathrm{P} 1=\operatorname{ROUND}[45.4] \quad(\mathrm{P} 1=45) \end{aligned}$ |
| Round up | CEIL [..] | $\mathrm{P} 1=\operatorname{CEIL}[8.3] \quad(\mathrm{P} 1=9)$ |
| Round down | FLOOR [..] | $\mathrm{P} 1=\mathrm{FLOOR}[8.7] \quad(\mathrm{P} 1=8)$ |

## Constants:

| $3.141592654(\pi)$ | PI | $\mathrm{P} 2=2 * \mathrm{PI}(\mathrm{P} 2=6.283185307)$ |
| :--- | :--- | :--- |

## Special functions:

| Check for existence of variables (V.P., V.L., V.S., V.E.)/ parameters / M/H functions / macros | EXIST [<Variable/ <br> Parameter/ <br> M function/ <br> H function/ <br> Macro>] | \$IF EXIST[V.P.MYVAR] == TRUE \$IF EXIST[V.L.MYARRAY[0]] == TRUE * <br> \$IF EXIST[P1] != TRUE <br> \$IF EXIST[M55] == TRUE <br> \$IF EXIST[H2O] == TRUE <br> \$IF EXIST["Macro1"] == TRUE <br> *For arrays with valid indices ! |
| :---: | :---: | :---: |
| Determine the size of an array <br> Dimension of variables (V.P., V.L.,V.S., V.E.) / parameters | SIZEOF [<Array_name>, <br> <Dimension>] <br> or for 1. Dim. <br> SIZEOF [<Array_name>] | $\begin{aligned} & \text { \$IF SIZEOF } \\ & {[V \cdot P \cdot M Y A R R A Y, 2]==3} \\ & \text { P1 }=\text { SIZEOF }[P 10,2] \end{aligned}$ <br> SIZEOF always results in -1 for non-existent array dimensions and for variable that are not arrays. |
| Determine smaller value | MIN $[x, y]$ | $\mathrm{P} 1=\mathrm{MIN}$ [P2, P3] |
| Determine greater value | $\operatorname{MAX}[x, y]$ | $\mathrm{P} 1=\mathrm{MAX}$ [P2, P3] |
| larger value sign | SIGN [..] | ```P1 = SIGN [P2] results in positive values: 1 negative values: -1 Zero: 0``` |

## Encryption functions:

These functions are used to encrypt and decrypt strings. The related key is user-definable. Strings may contain important data that require protection by encryption.

Encrypted data can then be saved to file with \#MSG SAVE, for example, or supplied to the PLC by V.E. variables.

| Encrypt string | ENCRYPT ["Key", "String" ] |
| :--- | :--- |
| Decrypt string | DECRYPT ["Key", "Decrypted string" ] |

The product of ENCRYPT or DECRYPT is assigned to a string type variable. In this case, note the following:

- With ENCRYPT, the string variable must at least be double the length of the string to be encrypted.
- With DECRYPT, the string variable must at least be half the length of the string to be decrypted.


## Programing Example

Encrypt string and save to file

```
N10 V.E.encrypted = ENCRYPT[ "Key", "String to be encrypted" ]
N20 #MSG SAVE ["Encrypted text = %s", V.E.encrypted ]
Decrypt encrypted string and output as message:
N10 V.E.decrypted = DECRYPT[ "Key", V.E.encrypted ]
N20 #MSG ["%s", V.E.decrypted ]
30
```


### 2.5.2 Operations for character strings

Overview of all available operations:
String operations:

| Adding strings | The + character combines 2 strings |
| :---: | :---: |
| + | ```V.E.str = "Hello" + " world!" (-> Result is "Hello world!")``` |


| Determine <br> left substring | LEFT supplies the left starting string of a string. Get anz characters <br> from string str based on the first character. <br> V.E.str $=$ LEFT ["Hello world!", 5] <br> $(->$ Result is "Hello") |
| :---: | :--- |
| LEFT[str, anz] | (-> |

## Determine

middle substring
MID supplies the substring of a string. Get anz characters from string str starting with character at position pos.

| MID[str, anz, pos] | V.E.str = MID["How are you?", 3, 5] <br> $(->$ Result is "are") |
| :---: | :--- |
| Determine <br> right substring RIGHT supplies the right final string of a string. Get anz characters <br> from string str based on the final character. <br> V.E.str = RIGHT["Hello world! How are you?", 12] <br> (-> Result is "How are you?") <br> RIGHT[str, anz]  |  | | (-> |
| :--- |


| Determining string length | LEN determines the length (number of characters) of a string. <br> LEN $=$ LEtr] $]$ <br> $(->$ Result is 25) |
| :---: | :--- |


| Determining character value | ORD supplies the numerical value of a character in a string at position pos. |
| :---: | :---: |
| ORD[str, pos] or ORD[str] | ```P1 = ORD["Hello world!", 1] (-> Result is 72, character value of "H") P2 = ORD["Hello world!", 7] (-> Result is 119, character value of "w")``` |
|  | If no position pos is specified, the value of the first character is returned. <br> If a position pos greater than the length of the character string is specified, the error ID 21545 is output. |
|  | For ASCII characters, the ORD function returns the exact ASCII value. |
|  | The return value for empty character strings is 0 . |
|  |  |

FIND[..] is case-sensitive (uppercase/lowercase).

| Search for substring | FIND searches for a string str2 in a string str1 and gives the position |
| :---: | :---: |
|  | of the first match of str2 in str1. |
| FIND[str1, str2] | V.E.str1 = "Hello world! How are you?" |
|  | V.E.str2 = "How" |
|  | P1 = FIND[V.E.str1, V.E.str2] (-> Result is 14) |
|  | does |
|  | V.E.str1 = "Hello world! How are you?" |
|  | V.E.str2 = "today" |
|  | P1 = FIND[V.E.str1, V.E.str2] |
|  | (-> Result is 0) |


| Deleting a substring | DELETE deletes in string str a specific number of characters anz, <br> starting with the character at position pos. |
| :---: | :--- |
| DELETE[str, anz, pos] | V.E.str = DELETE ["Hello world! How are you?", 5, <br> $7]$ <br> $(->$ Result is "Hello! How are you?") |


| Inserting a substring | INSERT inserts a string str2 in a string str1, starting with the character at position pos. <br> V.E.str1 = "Hello ! How are you?" <br> V.E.str2 = "world" <br> V.E.str = INSERT[V.E.str1, V.E.str2, 6] <br> (-> Result is "Hello world! How are you?") |
| :---: | :---: |
| INSERT[str1, str2, pos] |  |
|  |  |


| Replacing a substring | REPLACE replaces a number of characters anz in string str1 by the substring str2, starting with the character at position pos. |
| :---: | :---: |
| REPLACE[str1, str2, anz, pos] | V.E.str1 = "What is your name?" |
|  | V.E.str2 = "age" |
|  | ```V.E.str = REPLACE[V.E.str1, V.E.str2, 14]``` |
|  | (-> result is "What is your age?") |

## Comparison operators:

## Notice

Comparison operations are case-sensitive.

| Equality | V.E.str1 = "Peter" |
| :---: | :--- |
| $==$ | V.E.Str2 = "Peter" |
| \$IFV.E.str1 == V.E.str2 |  |
| \#MSG ["\%s is equal to \%s!", V.E.str1, V.E.str2] |  |
| \$ELSE <br> \#MSG ["Strings are not equal !"] <br> \$ENDIF <br> $(->R e s u l t ~ i s ~ " P e t e r ~ i s ~ e q u a l ~ t o ~ P e t e r ") ~$ |  |



| Greater than or Greater than or equal to | ```V.E.str1 = "Peter" V.E.str2 = "Peter" $IF V.E.str1 > V.E.str2 #MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2]``` |
| :---: | :---: |
| > | ```$ELSEIF V.E.str1 >= V.E.str2 #MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2] $ENDIF (-> Result is "Peter is greater than or equal to Peter!") V.E.str1 = "Peter" V.E.str2 = "Bob" $IF V.E.str1 > V.E.str2 #MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 >= V.E.str2 #MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2] $ENDIF (-> Result is "Peter is greater than Bob!")``` |


| Less than | V.E.str1 = "Peter" |
| :---: | :---: |
|  | V.E.str2 = "Peter" |
| Or | \$IF V.E.str1 < V.E.str2 |
| Less than or equal to | \#MSG ["\%s is less than \%s!", V.E.str1, V.E.str2] \$ELSEIF V.E.str1 <= V.E.str2 |
|  | \#MSG ["\%s is less than or equal to \%s!", V.E.str1, V.E.str2] |
| $<$ | \$ENDIF <br> (-> Result is "Peter is less than or equal to Peter!") |
| $<=$ |  |
|  | V.E.str1 = "Bob" |
|  | V.E.str2 = "Tim" |
|  | \$IF V.E.str1 < V.E.str2 <br> \#MSG ["\%s is less than \%s!", V.E.str1, V.E.str2] |
|  | \$ELSEIF V.E.str1 < V.E.str2 <br> \#MSG ["\%s is less than or equal to \%s!", V.E.str1, |
|  | V.E.str2] |
|  | \$ENDIF |
|  | (-> Result is "Bob is less than Tim!") |

## Conversion functions:

| Integer to String | INT_TO_STR[...] | V.E.str = INT_TO_STR[123] |
| :--- | :--- | :--- |
| Real to String | REAL_TO_STR[...] | V.E.str = REAL_TO_STR[12.34] |
| String to Integer | STR_TO_INT[...] | V.E.sgn32 = STR_TO_INT ["12"] |
| String to Real | STR_TO_REAL[...] | V.E.real64 = STR_TO_REAL["123.45"] |

### 2.5.3 Assigned address characters

Fixed meanings are assigned to the following address letters:
Address letters relating to technology:

| D,d | <int, double, expr> | Tool compensation call |
| :--- | :--- | :--- |
| E,e | <int, double, expr> | Feed at block end |
| F,f | <int, double, expr> | Feed at block start |
| H,h | <int, double, expr> | Auxiliary function |
| M,m | <int, double, expr> | Switch function |
| S,s | <int, double, expr> | Spindle speed, synchronisation ratio etc. |
| T,t | <int, double, expr> | Tool position call |

Address character relating to geometry:

| G,g | <int, double, expr> | Preparatory function |
| :--- | :--- | :--- |
| I,i | <int, double, expr> | Interpolation parameter for 1st path axis |
| J,j | <int, double, expr> | Interpolation parameter for 2nd path axis |
| K,k | <int, double, expr> | Interpolation parameter for 3rd path axis |
| R,r | <int, double, expr> | Circle radius |

## Address letters relating to program flow:

| L,I | <string> | Subroutine call, global |
| :--- | :--- | :--- |
| LL,II | <string> | Subroutine call, local |
| N,n | <int, double, expr> | Block number |
| O,o | <int, double, expr> | Not assigned |
| $\$$ |  | Identifier for control block statements |
| $\#$ |  | Identifier for extended syntax elements |

## Address letters relating to arithmetic:

| P,p | <int, double, expr> | Parameter |
| :--- | :--- | :--- |

Variabel und per Kanalparametersatz zuweisbar sind die Adressbuchstaben zur Bezeichnung der numerischen Achsen. Normally the letters $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ are used to designate the 3 linear axes of a Cartesian spatial coordinate system. A, B, C are used to designate rotary axes (case-sensitive). According to DIN 66025, the second motion in parallel to the spatial coordinate system is specified by U, V, W.

Besides this simple option, axis designations may also consist of several characters (strings) (X_ACHSE, Y22, ZA3). To distinguish between an axis designation and a coordinate value, the " $=\bar{"}$ character is used, i.e. $X 1=120.345$

The following sections in this programming manual mainly use the first case mentioned and often use address letters.

### 2.5.4 Programming examples

To clarify the subject mentioned above, a programming example is presented here in anticipation of the sections further below.

## The following address characters are used:

## Path preparatory functions:

G00 "Rapid traverse"
G01 "Linear interpolation"

## Spindle:

S1000 "Spindle speed 1000 rpm."

## Feedrate:

F5000 "Feedrate 5000 mm/min"

## Numerical axes:

X, Y, Z "three Cartesian axes"

## Machine functions:

M03 "Clockwise spindle rotation at programmed speed"
M05 "Spindle stop"
M30 "Program end"

## Programing Example

Summary of previous commands described

```
% 100
N10 G00 X100 Y100
N20 Z100;
N30 G1 Z50 F5000 S1000 M3
Z100;
N50 G0 X200 Y200 Z200
N50 M5
N60 M30
```

```
;Move at rapid traverse to X=Y=100
    ;Move at rapid traverse to Z=100 G00
    ;remains active until deselected by
    ;another G function occurs
    ;Clockwise spindle rotation 1000 rpm and
    ;Move linearly at feedrate
    ;5000 mm/min) to Z=5
    ;Move at feedrate to Z=100
;Move at rapid traverse to X=Y=Z=200
;Spindle stop
;Program end
```


## 3 Path information

### 3.1 Axis commands

Axis designations are configurable and must be taken from the configuration-specific description [1] [ 819]-5. When decoded, the notation is case-sensitive (uppercase/lowercase).

The following axis designations are available:

- Individual address letters: $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{U}, \mathrm{V}, \mathrm{W}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}, \mathrm{Q}\}$
- After programming an axis designation consisting of only one address letter, a space is required between the position value and the next character in order not to cause any mix-up when the equals sign is assigned afterwards.


## Example

The axis designations " X " and "X50P1" exist in the NC channel and axis " X " should be moved to position "50".

| X50P1 $=7$ | (ERROR) | X50P1 axis moves to position 7. |
| :--- | :---: | :--- |
| X50 P1 $=7$ | (RIGHT) | X axis moves to position 50. |

- Strings (e.g. X_SCHLITTEN, X1, Y22, Z_ACHSE)
- The first character of the string must correspond to one of the reserved address letters (see above). Other characters may include the numerals $0-9$. The string length of the axis designation may not exceed the maximum possible length (fixed), otherwise an error message is output.
To avoid confusion, an equals sign must be placed in front of the position statement after all axis designations consisting of more than one character.
This is necessary especially for axis designations ending with one of the numerals 0-9.


## Notice

X1 = <int, double, expr>

## Examples:

X1 $=100.0$
$\mathrm{X} 22=0,001$
X_SCHLITTEN = SIN [30]
Z_ACHSE = SQRT [2]/2

The following declarations also apply:

- All axis identifiers must be specified in the channel parameter block [1] [> 819]-5.
- An axis designation must always be followed by a numerical value or expression:


## X <int, double, expr>

## Examples:

X 100.0
Y 0.001
Z SIN [30]
A SQRT [2]/2
B 4 * R1/R2

## Programing Example

Axis commands

```
;Axis identifiers used:
;Y, Y50, Y_ACHSE_SCHL_1, Z7
N010 G01 F1500
N020 Y50 = 51 ;Axis Y50 to position 51
N030 Y52 ;Axis Y to position 52
N040 Y50 Z7 = 54 ;Axis Y to position 50 and
;Axis Z7 to position 54
N050 Y 70 Z7 = 55 ;Axis Y to position 70 and
;Axis Z7 to position 55
N060 Y = 71 Z7 = 56 ;Axis Y to position 71 and
;Axis Z7 to position 56
N070 Y[2+3] ;Axis Y to position 5
N080 Y50 = [4*3] ;Axis Y50 to position 12
N090 Y_ACHSE_SCHL_1 = 23 ;Axis Y_ACHSE_SCHL_1 to
;Position 23
N100 Y50 = P1 ;Axis Y50 to position P1
N110 M30
```

This programming manual uses the common designations $X, Y, Z$ for the three linear axes of a Cartesian system and $A$ and $B$ for two further path axes.

### 3.2 Measuring systems, input and precision ranges

The measuring systems used to specify positions, angles and velocities are the following:

Lengths and positions:
Linear velocities:
.. or also application-specific:
Angle:
Angular velocity:

$$
\begin{aligned}
& \mathrm{mm} \text { or inch } \\
& \mathrm{mm} \text { or inch per } \mathrm{s} \text { or min } \\
& \mathrm{m} \text { per min } \\
& \text { deg or gon, } \\
& \text { deg or gon per s or min. }
\end{aligned}
$$

In addition, the programmer has the freedom to apply self-defined measuring systems using parameter calculations.

All length specifications and linear velocities are calculated to a standard accuracy of $0.1 \mu \mathrm{~m}$. The maximum path length that can be entered at this resolution is then $-214 \mathrm{~m} . .+214 \mathrm{~m}$. Numerical inputs when programming may not exceed a range of -214000.0000 mm to +214000.0000 mm . This does not apply to individual elements (e.g. parameters) of an arithmetic expression if the result of the arithmetic expression is within the given range. It must be considered that the numerical range may not be exceeded even in combination with offsets and compensations.

Exception: The definition of a circle radius is possible up to maximum $10^{9} \mathrm{~mm}$. However, the target point of the arc must always be within the maximum path range $(-2.14 \ldots+2.14) * 10^{5} \mathrm{~mm}$.

Enter the axis type in the channel parameter block. If the application involves a rotary axis (or spindle), the angles are handled at a resolution of $0.0001^{\circ}$ by default. A range of $n * 360^{\circ}$ where $n=11900$ is then programmable.

At a resolution of $0.0001^{\circ}$, angles are possible within a range of
$(-2.14 \ldots+2.14) * 10^{5} / 360=2 * 594 \mathrm{rpm}$.

### 3.3 Coordinate systems

After the homing, the controller is at the machine zero point or in the machine coordinate system. If this is followed by inputs from the NC program (e.g. X100), the programmed coordinates (Index $p$ ) then coincide with the absolute coordinates (Index a):

$$
\begin{aligned}
& x_{\mathrm{a}}=x_{\mathrm{p}} \\
& y_{\mathrm{a}}=y_{\mathrm{p}}
\end{aligned}
$$

Offsets occur due to the definition of workpiece coordinate systems. The spatial positions in the workpiece coordinate system differ from the coordinate systems that are defined by the physical machine axes. Here a distinction must be made between programmed, constant, translatory offsets in a single axis and dynamic offsets that result from kinematic (e.g. cylindrical $\square$ Cartesian) or geometric transformations (e.g. tool radius compensation, mirroring), which in general affect several axes.

For example, the zero point can be offset from machine zero point $M$ to a freely selectable workpiece zero point $W$ or a workpiece coordinate system by a zero offset (NPV-G54...G59). The absolute coordinates result from adding zero offsets (NPV) and programmed coordinates:

$$
\begin{aligned}
& x_{\mathrm{a}}=x_{\mathrm{NPV}}+x_{\mathrm{p}} \\
& y_{\mathrm{a}}=y_{\mathrm{NPV}}+y_{\mathrm{p}}
\end{aligned}
$$

Irrespective of these NPVs specified by the zero offset data block, additional offset types can be explicitly programmed in the subroutine by using e.g. G92 X... Y ... Z ...
This reference point offset (BPV) is added to the preceding NPV. The absolute coordinates can then be determined as follows:

$$
\begin{aligned}
& x_{\mathrm{a}}=x_{\mathrm{NPV}}+x_{\mathrm{BPV}}+x_{\mathrm{p}} \\
& y_{\mathrm{a}}=y_{\mathrm{NPV}}+y_{\mathrm{BPV}}+y_{\mathrm{p}}
\end{aligned}
$$



Fig. 1: Defining a workpiece coordinate system using NPV and BPV (for legend, see below)

| $x a, y a$ | Absolute coordinates |
| :--- | :--- |
| $x p, y p$ | Programmed coordinates |
| $x N P V, y N P V$ | Zero offset |
| $x B P V, y B P V$ | Reference point offset |
| M: | Machine zero point |
| W: | Workpiece zero point |
| B: | Reference point for coordinates |
| P: | Position |

The coordinate display on the user interface shows active offsets by the remaining difference between the coordinates of physical machine axes (ACS) and workpiece coordinates (PCS). However, some offsets also result from manipulating machine and workpiece coordinates (e.g. tool radius compensation, mirroring) and therefore do not result in a coordinate difference.

The tables below provide an overview of the additional offset types in anticipation of the sections further below. The following conditions for the parameters are active:
activation and deactivation means the time when the offset is visible on the user interface because of coordinate differences or coordinate changes. However in general, an offset only becomes physically active at the earliest with the first motion that follows any activation or deactivation. For example, deactivation at program end leads to a compensating motion in the first motion of the following program.

Programmable offsets (linear, constant)

| No. | Description | Definition | ACS - <br> PCS <br> Differ- <br> ence <br> if <br> active | activation | deactivation | Temporary suppression |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | Reference point offset | NC program | Yes | NC block "G92 X.. Y.." G90/91 Consider dependency | NC block "G92 X0 Y0" or NC program start | "\#SUPPRESS OFFSETS" "\#MCS ON" |
| 2 | Zero offset | List NC prg. | Yes | $\begin{aligned} & \text { NC block } \\ & \text { "G54...G59" } \end{aligned}$ | NC block "G53" or NC program start | "\#SUPPRESS OFFSETS" "\#MCS ON" |
| 3 | Clamp position offset | List | Yes | Program paths Not changeable during program | NC program start Not changeable during program | „\#SUPPRESS OFFSETS" "\#MCS ON" |
| 4 | Tool offset | List NC prg. External | Yes | NC block "D.." | NC block "D0" or NC program start | "\#SUPPRESS OFFSETS" "\#MCS ON" |
| 5 | Position preset | NC program | Yes | NC block "\#PSET..." | NC block "\#PRESET..." or program start | "\#SUPPRESS OFFSETS" "\#MCS ON" |
| 6 | CS offset | NC program | Yes | NC block "\#CS ON[vx,vy,vz,.." | NC block "\#CS OFF" or NC program end | "\#MCS ON" |
| 7 | ACS offset | NC program | Yes | NC block "\#ACS ON[vx,vy,vz,.." | NC block "\#ACS OFF" or NC program end | "\#MCS ON" |

Offsets caused by geometric transformation (linear, dynamic):

| No. | Description | Definition | ACS - <br> PCS <br> Differ- <br> ence <br> if <br> active | activation | deactivation | Temporary <br> suppression |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 8 | CS | NC pro- <br> gram | Yes | NC block <br> "\#CS ON[.......]" | NC block "\#CS <br> OFF" <br> or | "\#MCS ON" |
| 9 | ACS | NC pro- <br> gram | Yes | NC block <br> "\#ACS ON[.......]" | NC block "\#ACS <br> OFF" <br> or | "\#MCS ON" |

Offsets caused by special functions:

| No. | Description | Definition | ACS - <br> PCS <br> Differ- <br> ence <br> if <br> active | activation | deactivation | Temporary <br> suppression |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 14 | Offset <br> due to <br> manual opera- <br> tion <br> with <br> parallel inter- <br> polation | Handwheel <br> NC prg. | Yes | NC block "G201" | NC block "G202" | Not <br> suppressible |
| 15 | Offset <br> due to <br> measurement <br> run | NC program | Yes | NC block "G101" | NC block "G102" | "\#SUPPRESS |
| OFFSETS" |  |  |  |  |  |  |

The NC command \#SUPPRESS OFFSETS only acts within an NC block
The NC command \#MCS ON deactivates any offsets until command \#MCS OFF is programmed.
Within every (A)CS the offset types 1, 2 and 5 are stored "locally".


Fig. 2: Overview of additional offsets and coordinate systems

## 4 <br> G functions

A complete list of G functions is contained in the overview of commands in the Appendix under G functions (G..) [〉 801].
"G" functions describe the type of feed motion, interpolation type and measuring method. They switch time-related influences and activate certain operation states. The syntax consists of the letter G combined with an identifier. The identifier is assigned a fixed validity and can be optionally preceded by a 0 :

## G <expr>

Various distinguishing features must be considered with G functions:

## Effectiveness:

There are G functions that only retain their validity for a specific block after programming (nonmodal) and $G$ functions which retain their validity the first time they are programmed until they are explicitly deselected modal).

## Exception:

Certain G functions mutually exclude each other. For example, G01 (linear interpolation) and G02 (circular interpolation) cannot be selected simultaneously. Therefore, the functions summarised in these groups may not be programmed in the same NC block.
A modal function is automatically deselected if a different function in the same group is selected in a following block.

## Programing Example

G functions

```
:
N90 G02 X100 Y50 I100
```

N50 G01 X100 Y200 ; linear interpolation effective
N60 G41 X200 Y200 ; linear interpolation effective
N70 X300 Y250 ; linear interpolation effective
N80 X100 Y50 ; linear interpolation effective
; circular interpolation effective

## Initial position:

During switching, after RESET or at program end, the controller is in initial position. In initial position, several $G$ functions are active without being explicitly selected.

### 4.1 Path preparatory functions

### 4.1.1 Rapid traverse (G00)

G00 Linear interpolation in rapid traverse $\quad$ (modal)

When G00 is selected, the rapid traverse velocity of the axes (specified in the machine parameters) is used for motion velocity. This results in an axis velocity where at least one axis is moved at its rapid traverse velocity.

Any number of straight lines can be programmed in the Cartesian spatial coordinate system (X, Y, Z). All programmed tracking axes move at linear velocity in such a way that the start and end of their motion take place simultaneously with the main axes.


Fig. 3: Position in rapid traverse with the parameters

## Programing Example

Rapid traverse G00

```
N05 G00 G90 X40 Y20 U30 ; move to starting point P1
; Absolute dimensional input:
N10 G00 G90 X120 Y80 U90 ; move from P1 to P2
; Incremental dimensional input:
N10 G00 G91 X80 Y60 U60 ; move from P1 to P2
```

Special case: Aligning a circular axis using G00 G90
If a rotary axis is programmed using G00 when G90 is active (G90: Absolute programming), the programmed target point is calculated in modulo, i.e. the rotary axis moves by a maximum of half a rotation.

### 4.1.2 <br> Linear interpolation (G01)

G01 Linear interpolation at programmed feedrate (modal)

If G01 is selected, the programmed path moves along a straight line to the target point at the feedrate specified in the F word (e.g. $\mathrm{mm} / \mathrm{min}$ ). Any number of straight lines can be programmed in the Cartesian spatial coordinate system (X, Y, Z). All programmed tracking axes move at linear velocity in such a way that the start and end of their motion take place simultaneously with the main axes.


Tracking axis U:


Fig. 4: Graphic display of linear interpolation (G01)

## Programing Example

## Linear interpolation G01

```
N05 G00 G90 X40 Y20 U30 ; move to starting point P1
; Absolute dimensional input:
N10 G01 G90 X60 Y60 U40 F1000 ; move from P1 to P2 feedrate 1000 mm/min)
N20 X120 Y80 U90 ;move from P2 to P3 feedrate 1000 mm/min)
; Incremental dimensional input:
N10 G01 G91 X20 Y40 U10 F1000 ; move from P1 to P2 feedrate 1000 mm/min)
N20 X60 Y20 U50 ;move from P2 to P3 feedrate 1000 mm/min)
```


### 4.1.3 <br> Circular interpolation (G02/G03)

Helical interpolation clockwise (CW)
G03
Helical interpolation counter-clockwise (CCW)

## (modal)

(modal)

When G02 or G03 is selected, the programmed path is travelled to the target point in circular motion at the feedrate specified in the F word. Circular motion can be travelled in the three main planes of the spatial coordinate system (X-Y, Z-X, Y-Z). The main plane is selected using the functions G17, G18, G19 (see Section: Plane selection [> 114]).

All programmed tracking axes move at linear velocity in such a way that the start and end of their motion take place simultaneously with the main axes.


Fig. 5: Description of circle functions G02 and G03
The circle is defined by taking the starting point of the circle "Ka" (determined in the preceding block), the target point of the circle "Ke" and the centre point of the circle "Km". The centre point of the circle is specified by the interpolation parameters I, J, K relative to the starting point of the circle when G162 is active or absolute when G161 is active.

| G162: | (Basic settings) |
| :---: | :--- |
| I | relative position of Km in the X direction |
| J | relative position of Km in the Y direction |
| K | relative position of Km in the Z direction |
| G161: |  |
| I | absolute position of Km in the X direction |
| J | absolute position of Km in the Y direction |
| K | absolute position of Km in the Z direction |

If the circle centre point is incorrectly defined, an error message is output if no centre point compensation is switched on (G165). When G165 is active, a centre point is defined so that a circle can be travelled. It also means that if the interpolation parameters are not programmed, circle centre point compensation originates at I, J, K = 0. In addition, the circle centre point coordinates are "non-modal".

## Notice

The maximum permissible circle radius is $10^{9} \mathrm{~mm}$. The target point of the arc may not exceed the maximum motion path of the axes of $+-2.14^{*} 10^{5} \mathrm{~mm}$.

If G02/G03 is active and if the interpolation parameters I, J, K are programmed with no circle end point, a full circle is travelled. Alternatively to I, J, K, the circle radius may also be defined with R (see Radius programming (R, G163) [ $\downarrow 140]$ ). However, it is not possible to program a full circle with R .

## Syntax example for G17 plane:

```
G02 | G03 [X<expr> Y <expr>] I<expr> J <expr> | R<expr>
```

G02 | G03
X<expr> Y<expr>
I<expr> J<expr>

R<expr>

Circular interpolation CW / CCW
Target point in XY plane in [mm, inch]
Position of circle centre point of interpolation in XY plane (I in X, J in Y) in [mm, inch], according to G161/G162
Radius of an interpolated partial circle (alternative to $I, J$ ) in connection with specification of a target point in [mm, inch].

## Syntax according to selected interpolation plane:

| Plane | Interpolation type | Target point in plane | Centre point/radius |
| :--- | :--- | :--- | :--- |
| G17 | G02/G03 | X..Y.. | I..J../R |
| G18 | G02/G03 | Z..X.. | K..I../R |
| G19 | G02/G03 | Y..Z.. | J..K../R |

## Programing Example

## Circular interpolation

```
N05 G0 X0 Y0
N10 G01 X10 Y10 F1000
N20 G02 X30 Y30 I10 J10 ;Semicircle about M1 circle end point X30 Y30
;Alternative N20:
N20 G02 X30 Y30 R[10*SQRT[2]] ;Semicircle about M1 circle end point X30 Y30
N30 G02 I10 J10 ;Full circle about M2
```



Alternatively, circles can also be programmed by specifying the radius. This is possible using G163="radius value" or using the address letter R="radius value". It is also possible to define using R1="radius value" (see Section Specifying centre point for circle definition (G161/G162) [ $>$ 139] - Controlling centre point offset in circle (G164/G165) [> 144]).

## Programing Example

## Circular interpolation

```
;Absolute dimensional input:
N05 G90 G00 X40 Y30 U40 ;Circle starting point (Ka), starting position
N10, G90 F1000 ;Absolute dimension, feedrate
N20 G17 ;Select X-Y plane
```

```
N30 G03 G161 X60 Y50 I60 J30 U90 ; Circle: Ka -> Ke and straight line: P1 -> P2
;Incremental dimensional input:
N05 G90 G00 X40 Y30 U40 ;Circle starting point (Ka), starting position
N10 G91 F1000
N20 G17
N30 G03 G162 X20 Y20 I20 U50
```

```
; Incremental dimension, feedrate
```

; Incremental dimension, feedrate
;Select X-Y plane
;Select X-Y plane
;Circle: Ka -> Ke and straight line: P1 -> P2

```
;Circle: Ka -> Ke and straight line: P1 -> P2
```

Main axes (here only X and Y ):



Fig. 6: Examples of circular interpolation

### 4.1.4 Helical interpolation (G02 Z.. K../G03 Z.. K..)

Helical interpolation is the superposition of a circular interpolation (plane of 1st and 2nd main axis) and a linear motion in the 3rd main axis. The resulting helical motion is executed at constant pitch. The pitch is programmed by the third parameter of the circular interpolation depending on the selected plane.


Fig. 7: Displaying helical interpolation at constant pitch

## Syntax example for G17 plane:

G02 | G03 X <expr> Y <expr> Z<expr> I<expr> J<expr> | $\mathbf{R}$ <expr> K<expr>

| G02 \| G03 | Circular interpolation CW / CCW |
| :--- | :--- |
| $\mathrm{X}<$ expr $>$ Y<expr $>$ | Target point in XY plane in [mm, inch] |
| $\mathrm{Z}<$ expr $>$ | Target point on helical axis perpendicular to XY plane in [mm, inch] |
| $\mathrm{I}<$ expr $>\mathrm{J}$ <expr> | Position of circle centre point of interpolation in XY plane (I in X, J in Y) in [mm, inch], ac- <br> cording to G161/G162 |
| $\mathrm{R}<$ expr $>$ | Radius of interpolated circle (alternative to I,J) |
| $\mathrm{K}<$ expr> | Pitch of helix in Z (value always without sign) in [mm, inch] |

Syntax according to selected interpolation plane:

| Plane | Interpolation <br> type | Target point <br> in plane | Target point <br> on helical axis | Centre point <br> /radius | Pitch |
| :---: | :---: | :---: | :---: | :---: | :---: |
| G17 | G02/G03 | X..Y.. | Z.. | I..J../R | K |
| G18 | G02/G03 | Z..X.. | Y.. | K..I../R | J |
| G19 | G02/G03 | Y..Z.. | X.. | J..K../R | I |

It is not necessary to define the pitch so that the helix reaches the programmed target point exactly. In this case, the NC kernel calculates a "corrected" pitch taking into consideration the fixed points for start and target points. The corrected pitch approximates to the programmed pitch as closely as possible.
In this case, the helix target point is first calculated based on the programmed pitch. If the calculated target point differs from the programmed target point, correction is required. The criteria for correction is the distance between the programmed target point and the calculated target point viewed in the direction of rotation.
If the distance is less than or equal to $\pi\left(180^{\circ}\right)$, the target point of the helix is shifted in the opposite direction of rotation towards the programmed target point, i.e. the pitch is increased.

If it is greater than $\pi\left(180^{\circ}\right)$, the target point of the helix is shifted in the direction of rotation, i.e. the pitch is reduced.


Fig. 8: Correcting the helix pitch depending on the direction of rotation

## Example

Principal correction of a helix clockwise (G02) (1st case)

The target point calculated using the programmed pitch $P_{\text {prog }}$ is within the range of $180^{\circ}$ after the programmed target point (viewed clockwise).
Correction takes place by reducing the pitch $\mathrm{P}_{\text {corr }}$.


Fig. 9: Correcting a helix within the range of $180^{\circ}$ after the programmed target point

## Example

## Principal correction of a helix clockwise (G02) (2nd case)

The target point calculated using the programmed pitch $P_{\text {prog }}$ is within the range of $180^{\circ}$ ahead of the programmed target point (viewed clockwise).
Correction takes place by reducing the pitch $\mathrm{P}_{\text {corr }}$.


Fig. 10: Correct a helix within the range of $180^{\circ}$ ahead of the programmed target point

## Programing Example

## Helical interpolation in the XY plane clockwise

The following helix is travelled:

| Starting point $a:$ | X-10 Y0 Z0 |  |
| :--- | :--- | :--- |
| Target point b: | X0 | Y-10 Z-20 |

Helix centre point I, J: Zero point
Helix pitch K: variable
:
N10 G17 G90 X-10 Y0 Z0 F500 G161
N20 G02 X0 Y-10 Z-20 I0 J0 K..


Fig. 11: Helical interpolation in the XY plane clockwise

Minimum rotation: $3 / 4 \rightarrow$ pitch $\mathrm{K}=26.66$

## Pitch $K \geq 26.66$ :

The helix from a to $b$ is generally executed in $3 / 4$ rotation because the correction is limited to the maximum possible pitch $\mathrm{K}=26.66$.

Pitch K < 26.666:

| Programmed <br> Pitch K (in mm) | Helix rotations from <br> a to b | Corrected <br> Pitch K (in mm) |
| :---: | :---: | :---: |
| 17.5 | $3 / 4$ | 26.66 |
| 16 | $13 / 4$ | 11.4 |
| 15 | $13 / 4$ | 11.4 |
| 12.5 | $13 / 4$ | 11.4 |
| 10 | $13 / 4$ | 11.4 |
| 7.5 | $23 / 4$ | 7.27 |
| 5 | $33 / 4$ | 5.33 |
| 2.5 | $73 / 4$ | 2.58 |
| 2 | $93 / 4$ | 2.05 |
| 1 | $193 / 4$ | 1.01 |

### 4.1.4.1 Simple helical interpolation

With simplified helical programming, no pitch is defined, only a target point. Depending on the target point the result is a helical motion with maximum one complete rotation.

## Syntax example for G17 plane:

| G02 \| G03 X <expr> Y <expr> Z<expr> I<expr> J<expr> \| $\mathbf{R}$ <expr> |  |
| :---: | :---: |
| G02 \| G03 | Circular interpolation CW / CCW |
| X<expr> Y<expr> | Target point in XY plane in [mm, inch] |
| Z<expr> | Target point on helical axis perpendicular to XY plane in [mm, inch] |
| I<expr> J<expr> | Position of circle centre point of interpolation in $X Y$ plane (I in $X, J$ in $Y$ ) in [mm, inch], according to G161/G162 |
| R <expr> | Radius of interpolated circle (alternative to I,J) in [mm, inch] |

Syntax according to selected interpolation plane:

| plane | Interpolation <br> type | Target point <br> in plane | Target place <br> on helical axis | Centre point/radius |
| :--- | :--- | :--- | :--- | :--- |
| G17 | G02/G03 | X..Y.. | Z.. | I..J../R |
| G18 | G02/G03 | Z..X.. | Y.. | K..I../R |
| G19 | G02/G03 | Y..Z.. | X.. | J..K../R |

## Programing Example

Helical interpolation in the XY plane counter-clockwise
The following helix is travelled:
Starting point a: $\quad \mathrm{X}-10$ Y0 Z0

Target point b: Z20
Helix centre point I, J: Zero point


Fig. 12: Helical interpolation in the XY plane counter-clockwise

### 4.1.5 Arc in space (G303)

## Release Note

This function is available as of CNC Build V3.01.3061.0

Arcs can be programmed in space using G303. An arc is clearly described by 3 points:

- Starting point
- Interim point
- Target point

Coordinates are programmed for:

- the interim point by I, J, K and
- for the target point by $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$

Starting point, interim point and target point of an arc may not be located on a straight line and the distance between each of the 3 points must be greater than 0 .

The 3 points define the arc plane. The motion direction is clearly fixed by the sequence starting point-interim point-target point.

The coordinates of the interim and target points refer to the currently active absolute or relative dimension specified (G90 or G91). With G91, the last point approached, i.e. the starting point, reference point.

## Syntax example of G303:

G303 I<expr> J<expr> K<expr> X<expr> Y<expr> Z<expr>
G303 Arc in 3D space
I.. J.. K.. Coordinates of interim point in space in [mm, inch]
X.. Y.. Z.. Coordinates of target point in space in [mm, inch]

## Restrictions:

- Full circles cannot be programmed using 3-point definition.
- Arc motion is only permitted if tool radius compensation (G40) is deselected.


## Programing Example

## Arc in space (G303)

```
%Circle_G303
;Start at X0 Y0 Z0
N10 G17 G90 G01 F2000 X0 Y0 Z0
;Move arc via interim point I,J,K (2) to target point X,Y,Z (3)
N20 G303 I30 J-15 K15 X60 Y0 Z30
N100 M30
```



Fig. 13: Arc in space defined by starting point (1), interim point (2) and target point (3)

### 4.1.6 Contour line programming (\#ANG)

In technical drawing, simple contours (e.g. of turned parts) are often described by specifying angles and points. This measurement is quick and easy to enter into an NC program using contour line programming.

Contour lines are located in a plane (G17, G18, G19) and describe a different type of linear block programming in the form of straight lines.
Contour line programming with active circular interpolation will result in an error message output.

## Contour line consisting of a straight line

Starting with a starting point (SP), a contour line describes the linear block by:

- Specifying an angle (ANG)
- and a coordinate (POS) of the target point (ZP).

The controller calculates the unknown second target point coordinate from the angle and the programmed coordinate. It is irrelevant which of the two target point coordinates is specified. Normally, this depends on the measurement in the actual drawing.

## Syntax example for G17 plane:

\#ANG $=$ <Angle> $\mathbf{X}<$ expr $>\mid \mathbf{Y}<$ expr $>\quad$ (modal)
\#ANG=<Angle>
X<expr>
Y<expr>

Angle relative to the first main axis of the active plane in [ ${ }^{\circ}$ ]
Coordinate of target point in the first main axis in [mm, inch]
Coordinate of target point in the second main axis in [mm, inch]

## Contour line with coordinate in the first main axis



Fig. 14: Contour line with coordinate in the first main axis

## Programing Example

Contour line in G17 with target coordinate in X

| N10 G17 G90 G0 X10 Y10 |  | ; Approach start position |
| :--- | :--- | :--- | :--- | :--- | :--- |
| N20 G01 F2000 \#ANG=60 X20 | ; Contour line to target point X20, Y27.3205 |  |
| N30 ... |  |  |

Contour line with coordinate in the second main axis


Fig. 15: Contour line with coordinate in the second main axis

## Programing Example

Contour line in G17 with target coordinate in $Y$

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=35 Y20 ; Contour line to target point X24.2812, Y20
N30 ...
```



Page break
Validity check of target point:
When the complete target point is determined, a check is made whether the programmed target point coordinate (POS) can be reached with the specified angle. If the target point cannot be reached with the specified angle, an error message is output.


Fig. 16: Validity range of target point

## Contour line consisting of 2 straight lines

A contour line consisting of 2 straight lines can be programmed in various combinations of angles and target coordinates. The associated rules are explained in the permitted cases listed below.
Page break

## Case 1: Combination of two angles and two target coordinates

The target point ZP1 of the first straight line results from an angle ANG1 and a target coordinate. Based on this, the target point ZP2 of the second straight line also results from an angle ANG2 and a target coordinate. The target coordinates of ZP1 and ZP2 can be programmed as absolute (G90) or relative (G91).


Fig. 17: Contour line with two straight lines (2 angles each with one target coordinate)

## Programing Example

Contour line with 2 straight lines in G17 and 2 angles with target coordinates

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=25 X30 ; Straight line 1, target point: X30, Y19.3258
N30 #ANG=120 Y50 ;Straight line 2, target point: X12.2904, Y50
N40 ...
```



## Case 2: Combination of two angles and target coordinate 2

The angles ANG1 and ANG2 are each programmed completely (Cartesian) for the two straight lines and for the second straight line of the target point ZP2. The target point ZP2 must always be specified as absolute (G90). The target point of the first straight line ZP1 can then be determined as the intersecting point of the straight lines.


Fig. 18: Contour line with 2 straight lines, 2 angles, complete target point 2

## Programing Example

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=15 ; Straight line 1
N30 #ANG=100 Y60 ; Straight line 2, target point 2:
N40 ...
```



## Special case 2-1: Combination of two angles and one target coordinate 2

The angles ANG1 and ANG2 are each programmed for the two straight lines and only one target coordinate of ZP2 for the second straight line. The other coordinate of target point ZP2 results from the associated components of starting point SP. The target coordinate ZP2 must always be specified as absolute (G90). The target point of the first straight line ZP1 can then be determined as the intersecting point of the straight lines.


Fig. 19: Contour line with 2 straight lines, 2 angles, incomplete target point 2

## Programing Example

```
N10 G17 G90 G0 X10 Y10 ;Approach start position
N20 G01 F2000 #ANG=75 ; Straight line 1
N30 #ANG=100 Y60 ; Straight line 2, one target point
N40 ...
```



## Special case 2-2: Combination of two angles, no target coordinates

If only angles and no target coordinates are programmed, the target points ZP1 and ZP2 are identical to the starting point SP. Only two motions are then possible perpendicular to the current plane.

## Validity check of target points:

This checks whether the programmed target points can be reached with the programmed angle starting from the starting point. The orientations resulting from the programmed angles define the valid range for the target points.


Fig. 20: Validity range of target points with 2 straight lines

## Contour line consisting of several straight lines

Any number of straight lines can be connected together to describe a contour. The target points of the straight lines must then be clearly determinable geometrically. The programming rules for a contour line with 2 straight lines must also be complied with for linked contours.

## Programing Example

## Contour line with several straight lines in G17

```
N020 G17 G90 G01 F2000
N030 X10 Y10 ; Move to start position
NO40 #ANG=0 X20
N050 #ANG=90 Y20
N060 #ANG=45 X30
N070 #ANG=135 X20
N080 #ANG=90 Y50
N090 #ANG=180 X15
N100 #ANG=135 Y55
N110 #ANG=225 Y50
N120 #ANG=180 X0
N130 #ANG=270 Y40
N140 #ANG=225 Y30
N150 #ANG=315 Y20
N160 #ANG=270 Y10
N170 #ANG=0 X10
N180 M30
```



## Contour lines in combination with chamfers and roundings

Contour lines can be combined with a complete scope of functions for programming chamfers and roundings (see Section Chamfers and roundings [ 158]). This is illustrated by the programming example for a turned part below.

## Programing Example

Contour line of a turned part with chamfers and roundings

```
N030 G18 G90 G00 X0 Z150
N040 X5 G01 F2000
N050 #ANG=100 #CHR=5 #FRC=1000
N060 #ANG=130 X25 Z140 #RND=5 #FRC=1500
N070 #ANG=90 X40 #CHR=4 #FRC=1000
N080 Z120 #RND=5 #FRC=1500
N090 #ANG=140 X50 #CHR=2 #FRC=1000
N100 Z100
N110 ...
```



Z

### 4.1.7 Dwell time (G04), (\#TIME)

G04 <1.Hauptachse><expr> | <expr> | <Hauptspindel><expr> (non modal)

## G04

<1.Hauptachse><expr> <expr>
<Hauptspindel><expr>

## Dwell time

Dwell time is specified by the name of the 1.Hauptachse in [s] or, alternatively...
... as following direct or parameterised specification [starting with Build V2.11.2026.02] of the dwell time in [s] or ...
By the name of the Hauptspindel and by specifying a number of revolutions [U] [as of V2.11.2023.02]

Dwell times are required for relief cutting or other machine functions, for example.
Dwell time may only be programmed in the NC block alone (exception: block no.).

## Programing Example

Dwell time (G04)

```
N10 G04 X4.5 (wait 4.5 seconds)
N20 G04 3.0 (wait 3.0 seconds)
N30 P1=2
N40 G04 P1 (wait 2.0 seconds)
N50 V.L.TIME=3.5
N60 G04 V.L.TIME (wait 3.5 seconds)
N70 M3 S200
N80 G04 S10 (wait 10 revolutions (wait 3 seconds)
```

Another possibility to specify dwell time is to use the function \#TIME.
\#TIME <expr> (non modal)

| \#TIME | Dwell time plain text command |
| :--- | :--- |
| <expr> | Dwell time value direct or parameterised in [s] |

## Programing Example

```
N10 #TIME 2.5
N20 P1=2
N30 #TIME P1 (wait 2.0 seconds)
N40 V.L.TIME=3.5
N50 #TIME V.L.TIME (wait 3.5 seconds)
```


### 4.1.8 Programmable homing (G74)

G74 Approach reference point (non-modal)

G74 allows the NC program-controlled execution of a homing motion (RPF) which must contain the statements on the axes to be referenced and the sequence in which the axes are to execute the homing motion. The values programmed with axis names define the homing sequence.
For axes with the same value, homing is triggered simultaneously.
Further information on referencing is contained in the functional description of "Homing" [FCTM1].

## Programing Example

Programmable homing (G74)

```
;Sequential commanding:
N10 G74 X2 Y3 Z1 ;Homing sequence: Z-> X -> Y
;Parallel commanding:
N10 G74 X1 Y1 Z1 ;Homing sequence: X,Y,Z simultaneously
```


### 4.1.9 Reference point offset (G92)

G92 Reference point offset (non-modal)

G92 allows a programmable reference point offset in the given axes by a freely programmable value in [mm, inch] (additive reference point offset). Depending on the G90/G91 setting, the currently programmed reference point offset is set absolutely or added to the existing one.

```
Notice
"Non-modal"
...only applies to G92; of course, reference point offset itself only applies up to the new G92 programming
```


## Programing Example

Reference point offset (G92)

| N10 G90 | (Absolute dimensional specification) |
| :--- | :--- |
| N20 G92 X10 z30 | (Displaces the programmed and the absolute) <br> (coordinates by 10 in $X, 30$ in $Z)$. |
| . |  |
| . |  |

## Notice

i
In addition, the following applies to selecting a reference point offset in G91 mode:
The programming of...
N10 G92 X10 Y20
N20 G0 X0 G91
... may not cause any movement of the X axis (corresponds to relative movement about 0 ). The reference point offset is then only effective for an axis when the next motion information is programmed in absolute mode (G90).

### 4.1.10 Set negative software limit switch (G98)

G98 Set negative software limit switch (non-modal)

G98 sets the negative limit switch positions in [mm, inch] in all programmed axes. Depending on the G90/G91 setting, this may be absolute or additive to the previous software limit switch position.
The positions for negative limit switches are saved in the axis-specific variables:
V.A.-SWE.X, V.A.-SWE.Y, V.A.-SWE.Z, etc.
(see also Section Axis-specific variables [> 575]).

## Notice

"Non-modal"
...only applies to the command G98; the software limits switches themselves are effective in modal mode.

After machine start-up, the default value of axis parameter P-AXIS-00177 is valid first.
The following applies concerning the validity of limit switch positions for all builds of V 2.11 .20 xx and $\mathrm{V} 2.11 .28 x \mathrm{x}$ :

- The limit value can be further restricted in the NC program by programming but cannot be increased. In other words, the limit value defined in the axis parameter list cannot be increased by G98.
- In static axis constellations (without axis exchange) the limit value changed in the NC program first remains valid at program end and is also effective in the next NC program activated. Only after CNC reset followed by a program restart does the original default value become valid again.
As of Build V3.1.3077.0 the following applies:
- See description in Section "Supplements to G98 and G99"

In dynamic axis constellations (with axis exchange) reset to the original default value is executed when the axis is adopted in the channel.

A G98 change acts on the axis motion path range, on independent axes and on single axes. Relative motion path ranges in manual mode are not affected; they are influenced by the NC command \#MANUAL LIMITS [...] [ 174].

## Programing Example

## Set negative software limit switch (G98)

```
N10 G90
```

N100 G98 X-1000 Y-2000 Sets negative software limit switch
in $X$ to -1000 and in $Y$ to -2000


### 4.1.11 Set the positive software limit switch (G99)

G99 Set the positive software limit switch (non-modal)

G99 sets the positive limit switch positions in [mm, inch] in all programmed axes. Depending on the G90/G91 setting, this can be absolute or additive to the previous software limit switch position.

The positions for positive limit switches are in the axis-specific variables
V.A.+SWE.X, V.A.+SWE.Y, V.A.+SWE.Z, etc.
(see also Section Axis-specific variables [> 575]).

Notice
"Non-modal"
...only applies to command G99; the software limit switches themselves are effective in modal mode.

After machine start-up, the default value of axis parameter P-AXIS-00178 is first valid.
The following applies concerning the validity of limit switch positions for all builds of V2.11.20xx and V 2.11 .28 xx :

- The limit value can be further restricted in the NC program by programming but cannot be increased. In other words, the limit value defined in the axis parameter list cannot be increased by G98.
- In static axis constellations (without axis exchange) the limit value changed in the NC program first remains valid at program end and is also effective in the next NC program activated. Only after CNC reset followed by a program restart does the original default value become valid again.

As of Build V3.1.3077.0 the following applies:

- See description in Section "Supplements to G98 and G99"

A G99 change acts on the axis motion path range, on independent axes and on single axes. Relative motion path ranges in manual mode are not affected; they are influenced by the NC command \#MANUAL LIMITS [...] [> 174].

## Programing Example

## Set the positive software limit switch (G99)

```
Sets positive software limit switch
in X to +1000 and in Y to +2000
```



### 4.1.12 Extensions to G98 and G99

As of Build V3.1.3077.0 limit switch positions referenced to the limits specified in the axis parameter list can also be extended by G98 and G99. This permits a temporary change to an extended section within an NC program and back. The positive limits must continue to be greater than the negative limit. At the next NC program started or after a CNC reset, the configured default values will again a0ppOly.

The following applies in

- automatic mode: When G98/G99 are programmed before the motion movement, the limited section compared to the configured setting can also be enlarged.
- Manual mode: When G98/G99 are programmed before manual mode is activated,, the limited section compared to the configured setting can be extended:
- Relative offset limits P-AXIS-00137 and P-AXIS-00138. The new limit acts immediately when manual mode is activated.
- Absolute offset limits P-AXIS-00492 and P-AXIS-00493: These values are effective if they are entered in the axis parameter list !=0. They can then be set to the maximum configured limit switch positions (P-AXIS-00177, P-AXIS-00178). In this way, the axis cannot move beyond these limits in manual mode, even if the limits are extended by G98 and G99. However, it is possible to adapt these absolute manual mode limits to the new limit switch positions using CNC variables.


## Programing Example

Increase software limit switch range using G98 and G99

```
;Assuming: Software limit switches are configured to +- 200 in X, Y
N10 G01 G90 X199
N100 G98 X-500 Y-500 ; neg. Software limit switches X and Y -> -500
N200 G99 X500 Y500 ;pos. Software limit switches X and Y -> +500
N300 G01 X450 Y450 ;Move within extended section
N400 G01 X100 Y100 ;Back to limited section
N500 G98 X-200 Y-200 ; neg. Software limit switches X and Y -> -200
N600 G99 X200 Y200 ;pos. Software limit switches X and Y -> 200
```


### 4.1.13 Measuring functions

After controller start-up, the measurement type specified in the channel parameter P-CHAN-00057 is specified. In the NC program, \#MEAS MODE [> 348] or \#MEAS [TYPE..] [ 349] can be used at any time to select a new measurement type.

In the NC program, the variable V.G.MEAS_TYPE [ 584] supplies the currently valid measurement type. The following 7 measurement types are available:

| Measurement <br> type | Meaning |
| :--- | :--- |
| $1^{*}$ | Measurement run with at least one axis, <br> measuring feed programmable via F word. |
| $2^{*}$ | Measurement run with exactly one axis, <br> measuring feed is defined in P-AXIS-00215. |


|  | An error message is output if the probe signal is missing. |
| :--- | :--- |
| 3 | Measurement run with at least one axis, <br> measuring feed programmable via F word, <br> optionally with continued motion up to the target point. |
| 4 | Measurement run only with maximum 3 main axes, <br> measuring feed programmable via F word. |
| 5 | Interruptible measurement run with at least one axis, <br> measuring feed programmable via F word. |
| 6 | Interruptible measurement run with at least one SERCOS axis, <br> measuring feed programmable via F word. |
| $7^{*}$ | Measurement run with motion to a fixed stop with at least one axis, measuring <br> feed programmable via F word. |

* with these measurement types, a measurement run is also possible using independent axes [ 774].

The following variables related to the measurement run are available in the NC program (see also Section Axis-specific variables (V.A) [ 575] ).

## Notice

Only the variable values of the axes programmed during the measurement run are updated. All other axis-specific measuring variables retain the old values or 0 .

| V.A.MERF.<axis> | Measurement run completed? |
| :--- | :--- |
| V.A.MESS.<axis> | Supplies the axis-specific measuring value in [mm, inch] after a <br> measurement run in which the measurement took place. The value <br> always includes all offsets in the calculation.. |
|  | With 2.5D: ACS values or with CS / TRAFO: PCS values |
| V.A.MOFFS.<axis> | Measuring offset in [mm, inch] |
| V.A.MEIN.<axis> | Measuring offset calculated in [mm, inch] |

## Release Note

As of Build V2.11.2020.07
V.A.MEAS.ACS.VALUE.<axis>
V.A.MEAS.PCS.VALUE.<axis>

Measurement value in the axis coordinate system in [mm, inch] including all offsets.
Measured value in the coordinate programming system in [mm, inch] without offsets.

## Restrictions:

A measurement run cannot be programmed if:

- Polynomial contouring (G261, G61) is active
- Spline interpolation (AKIMA, B-Spline) is active
- HSC functions are active


### 4.1.13.1 Measure with multiple axes (G100) (Type 1)

G100 <axis_name><expr> \{<axis_name><expr>\} [F<expr>] (non-modal)

| G100 | Select measurement run |
| :--- | :--- |
| <axis_name><expr> | Target point of measuring axis in [mm, inch] |
| F<expr> | Measuring feed in $[\mathrm{mm} / \mathrm{min}, \mathrm{m} / \mathrm{min}, \mathrm{inch} / \mathrm{min}]$ |

Any axis may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 1) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. At least one axis must participate in a measurement run. The measuring feed is specified by the F word. The motion path in the measuring block must be greater than 0 . Programing Example
Measure with multiple axes (G100) (Type 1)
General representation of a measurement run.
\%G100_Type_1
N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 ; X10/Y20 target point of the measurement run
...

$d_{m}$ : Measuring traverse path
Fig. 21: Program the measuring function Type 1

The program stops after the probe signal is detected. The remaining motion path of the measuring block is no longer output.

## Programing Example

## Measure with multiple axes (G100) (Type 1)

Successful measurement run followed by continuing along programmed path.

```
%Meas_run
N10 G90 G00 X0 Y0 Z0
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
N50 M30
```



Fig. 22: Programmed path with measuring function Type 1

### 4.1.13.2 Measure with a single axis (G100) (Type 2)

G100 <axis_name><expr> (non-modal)

| G100 | Select measurement run |
| :--- | :--- |
| <axis_name><expr> | Target point of measuring axis in [mm, inch] |

Only one axis may participate in the path motion of the measuring block. The axis programmed in the measuring block must be identified as the measuring axis (P-AXIS-00118). The measuring method (Type 2) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. Exactly one axis must participate in a measurement run. The measuring feed is specified in P-AXIS-00215. The motion path in the measuring block must be greater than zero.

## Programing Example

## Measure with a single axis (G100) (Type 2)

General representation of a measurement run


## $\mathrm{d}_{\mathrm{m}}$ : Measuring traverse path

Fig. 23: Program the measuring function Type 2

The program stops after the probe signal is detected. The remaining motion path of the measuring block is no longer output. An error message is output if no probe signal is adopted in the measuring block.

## Programing Example

## Measure with a single axis (G100) (Type 2)

Successful measurement run followed by continuation along programmed path.

```
%Meas_run
N10 G90 G00 X0 Y0 Z0
N20 Y5
N30 G100 X10
N40 G01 X7
N50 M30
```



Fig. 24: Programmed path with measuring function Type 2
kernel Industrielle Steuerungstechnik GmbH

### 4.1.13.3 Measure with motion through to target point (G100/G106) (Type 3)

```
G100 <axis_name><expr> { <axis_name><expr> } [G106] [F<expr> ]
```

G100 Select measurement run
<axis_name><expr>
G106
F<expr>
Select measurement run
Target point of measuring axis in [mm, inch]
Move up to target point
Measuring feed in [mm/min, m/min, inch/min]

```

Any axis may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 3) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. At least one axis must participate in a measurement run. The measuring feed is specified by the F word. The motion path in the measuring block must be greater than 0 .

Programing Example
Measure with motion up to target point (G100/G106) (Type 3)
General representation of a measurement run
```

%G100_Type_3
N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 G106 ;X10/Y20 target point of the measurement run
...

```

\(d_{m}\) : Measuring traverse path
Fig. 25: Program the measuring function Type 3

After a probe signal is detected, the system continues up to the target point of the measuring block if G106 is programmed. If G106 is not programmed, the system decelerates after the probe signal and the remaining motion path is no longer output (same reaction as with measurement type 1).

\section*{Programing Example}

\section*{Measure with motion through to target point (G100/G106) (Type 3)}

Successful measurement run followed by continuation along programmed path.

\footnotetext{
\%Meas_run
}

N10 G90 G00 X0 Y0 Z0
N20 G01 X5 F500
N30 G100 G106 X10 Y10 ; After probe signal, move up to target point
N40 G01 X7
N50 M30


Fig. 26: Programmed path with measuring function Type 3

\subsection*{4.1.13.4 Measure with main axes (G100) (Type 4)}

G100 <axis_name><expr> \{<axis_name><expr> \} [F<expr>] (non-modal)
\begin{tabular}{ll} 
G100 & Select measurement run \\
<axis_name><expr> & Target point of measuring axis in [mm, inch] \\
F<expr> & Measuring feed in \([\mathrm{mm} / \mathrm{min}, \mathrm{m} / \mathrm{min}, \mathrm{inch} / \mathrm{min}]\)
\end{tabular}

The three main axes may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 4) must be parameterised (P-CHAN-00057).

During measurement, the receipt of a probe signal is detected in the measuring block. Linear interpolation is performed between the target point specified in the NC command and the starting point (same effect as with G01). Below, the path velocity in the measuring block is referred to as 'measuring feed'. A maximum of three main axes may participate in a measurement run. The measuring feed is specified by the F word. The motion path in the measuring block must be greater than 0 .

\section*{Programing Example}

Measure with main axes (G100) (Type 4)
General representation of a measurement run.
\%G100_Type_4
N10 G90 G00 X0 Yo
N20 G100 X10 Y20 F200 ; X10/Y20 target point of the measurement run

\(\mathrm{d}_{\mathrm{m}}\) : Measuring traverse path

Fig. 27: Program the measuring function Type 4

The program stops after the probe signal is detected. The remaining motion path of the measuring block is no longer output.

\section*{Programing Example}

\section*{Measure with main axes (G100) (Type 4)}

Successful measurement run followed by continuation along programmed path.
```

%Meas_run
N10 G90 G00 X0 Y0 Z0
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
N50 M30

```


Fig. 28: Programmed path with measuring function Type 4

\subsection*{4.1.13.5 Measure with interruption and jump (G310) (Types 5, 6)}
```

G310 [G00 | G01 F<expr>] <axis_name><expr> {<axis_name><expr>} [\$GOTO<Label>] (non-modal)

```
\begin{tabular}{ll} 
G310 & Interruptible block \\
G00 | G01 & Interruptible interpolation modes \\
F<expr> & Measuring feed in [mm/min, m/min, inch/min] \\
<axis_name><expr> & Measuring axes with target points in [mm, inch] \\
\$GOTO<Label> & Jump target after interrupted measurement run
\end{tabular}

Any axis may participate in the path motion of the measuring block. All axes programmed in the measuring block must be identified as a measuring axis (P-AXIS-00118). The measuring method (Type 5.6) must be parameterised (P-CHAN-00057).

This measuring method offers the option to abort a motion by a probe signal. The path motion must be explicitly programmed in the same block. When the path motion is aborted by the probe signal, the program branches to the jump target (label) specified in the G310 block. If the probe signal does not occur during the motion block, the NC program is continued with the next NC block.

\section*{Programing Example}

\section*{Measure with interruption and jump (G310) (Types 5, 6)}
```

N10 G00 X0 Y0
N20 G310 G01 F100 X100 Y200 \$GOTO[N_LABEL]
;If interrupted, jump to N_LABEL
N30 G01 X200
N40 \$GOTO[ENDE]
N50 [N LABEL] X0 Y0
N60 [ENDE] M30

```

After the motion is interrupted by a probe signal, the coordinates of the programmed target point are replaced by the actual positions of all the measuring axes in the channel. Next, the logic jumps to the specified block
If no signal is received, the system moves to the programmed target point. So, then there is no jump and the next block is executed instead.

The next block is always executed if a jump target (label) was not programmed.
The current axis positions after an interruption in path motion can be read in the NC program using the V.A.MESS[..] variable.

If G310 is programmed when TRC (G41/G42) is active, the program is interrupted and an error message is output.

\subsection*{4.1.13.6 Measure with motion to a fixed stop (G100) (Type 7)}

When a measurement run is executed with motion to a fixed stop, torque limitation must be activated in all drives involved and any drive-based following error monitor must be disabled.
The measurement run ends as soon as the fixed stop is detected in one of the axes involved in the measurement run.

A programming example and the settings required for a measurement run with motion to a fixed stop are described in greater detail in the functional description "Measurement (C4)" ([FCT-C4]).

\subsection*{4.1.13.7 Calculate measuring offsets (G101/G102)}

The measuring offset is the offset depicted in the figure below between the recorded measuring position and the target point. It is calculated as follows:

Measurement offset \(=\) measuring point \(\boldsymbol{-}\) target point


Fig. 29: Measurement offset between probe position and programmed target position
G101 <axis_name><fact> \{<axis_name><fact> \} (non- modal)

G101
<axis_name><expr>

Include measuring offset calculation in offset
Axis-specific factor to include measuring offset in calculation

\section*{Notice}

Several axes can also be specified for measurement type 2 if a separate measurement run was conducted for each of these axes beforehand.

For the programmed coordinates, the measuring offset determined from the measured values is included in the calculation of an additional offset between programmed and absolute coordinates. An error message is output if no measured values were detected beforehand. The numeral after the axis designation represents the inclusion factor.

The offset caused by the measurement offset is valid until it is deselected by G102.

\section*{Programing Example}

\section*{Calculate measuring offsets (G101/G102)}

Includes the measuring offset for \(X\) in the calculation with factor 1 and for \(Y\) with factor 7 in the offset between programmed and absolute coordinates.

G102 \{<axis_name><dummy_expr>\} (non-modal)
\begin{tabular}{ll} 
G102 & Extract measuring offset from offset \\
<axis_name><dummy_expr> & Axis for which the measuring offset is extracted
\end{tabular}

The measured values adopted with G100 and included in the calculation as further offsets with G101 are extracted according to the following rule:
If one or more axes are programmed, only these axes are calculated. If no axis was programmed, all offsets are extracted.
An error message is generated if an axis was programmed for which no measuring offset is included in the calculation.

Compared to G101, the value behind the axis specified is meaningless but requested due to syntactical reasons. The offset included in the calculation with G101 is always extracted.

\section*{Programing Example}

Calculate measuring offsets (G101/G102)
Extract an axis/all axes
```

N10 G102 X1 ;Only extract offset of X-axis
N20 G102 ;Extract offsets of all axes

```

\subsection*{4.1.13.8 Edge banding (G108)}

The wood machining and furniture industry requires the edge banding function to perform the exact glueing of veneer strips. When a veneer strip is glued on, the position of the edge start may differ on various workpieces by several millimetres. So, the veneer strip must be cut at a position that is different from the NC block limit. This is a two-dimensional problem in the XY plane and linear and circular motion is allowed.

The start of the veneer strip is measured by a leading probe (see Fig. below). The measured values are used to determine the exact position at which the veneer strip must be cut.

The following response must be achieved:
When the measuring probe is crossed, the interpolator outputs a specified distance to go. The system then waits until all axes are inside the control window. An M function of synchronisation mode MNE_SNS (P-CHAN-00027) is then transferred to the PLC interface to cut the veneer strip. After acknowledgement, interpolation is continued up to the programmed target point. Only one MNE_SNS function may be active, i.e. multiple programming in one NC block is not allowed. But the MNE_SNS function may be programmed in a block using the \(M\) functions of other synchronisation modes.


Fig. 30: Glue on a veneer strip

Edge banding is selected using the G108 function which is modal. Depending on the parameterisation in the channel parameter block P-CHAN-00029, the following 2 methods are possible:

\section*{Notice}

If only one CNC axis is moved, G108 is also allowed for the special case without main axis motion. In this case the measurement may take place in one of the tracking axes.

\subsection*{4.1.13.8.1 Glue in one motion block (Method 1)}

Selecting the measurement using G108 has a modal effect until an M function of synchronisation mode MNE_SNS (P-CHAN-00027) is programmed. The measurement run is started with this M function. The probe signal must occur in the motion block that was programmed together with this M function.

After the probe signal is detected, the system continues moving along the distance to go P-CHAN-00030 and G108 is deactivated implicitly.

\section*{Programing Example}

Glue in one motion block (Method 1)

N05 X0 Y0
N10 G108 (Activate edge banding)
N20 G01 X90 Y90 F20
N30 M97 G01 X150 Y150 F8

M30
(M97 is of mode MNE_SNS, start measurement run)
(Continue with distānce to go after probe signal)

\section*{Attention}

If no probe signal is detected during the measurement run, an error message is output at block end.

\subsection*{4.1.13.8.2 Edge banding across several motion blocks (G107)(Method 2)}

The problem often involves a contour which is described by several short NC blocks (e.g. generated by a CAD system). If edge banding is to occur on this type of contour, the resulting problem is to hit the edge exactly on a contour element.

G108 [ \(>\) 104] also activates edge banding and has a modal effect.
Special parameterisation in the channel parameter block (P-CHAN-00029) offers the extended function to only execute edge banding \(n\) blocks after the M function. The M function itself is also output immediately but the trigger event is only output later. After the probe signal is detected, the system continues moving along the distance to go for a further m blocks. The NC command G107 indicates explicitly the latest point at which the measurement run must end.
\begin{tabular}{|lll|}
\hline G107 & Deselect cross-block edge banding & (modal) \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Glue across several motion blocks (Method 2)}
```

N05 X0 Y0
N10 G108 (Activate edge banding)
N20 G01 X90 Y90 F20
N30 M97 G01 X100 Y100 F8 (M97 is of mode MNE SNS,start measurement run)
N40 X110 Y110
N50 X120 Y120
N60 X130 Y130
N70 X140 Y140
N80 X150 Y150 (<- last measurement run block!)
N90 G107 (End of measuring edge banding)
N80 G00 X200 Y200
M30

```

\section*{Notice}

The measuring process is also possible in the same motion block. However, method 1 offers simpler programming for this case.

Attention
If no probe signal is detected during the measurement run, an error message is output with G 107 .

\subsection*{4.1.13.8.3 Program distance to go}
V.G.RW Distance to go with edge banding (Read and write access)

The axis group-specific "V.G.RW" variable can be used to redefine the distance to go for edge banding in [mm, inch]. The distance to go is the distance remaining after the probe signal is received. In initial state, the corresponding working data point for the distance to go is adopted from the channel parameter block (P-CHAN-00030).


Programing Example
Program distance to go
```

N040 G01 X10 Y10 F300
N050 M97 G01 X30 Y10 F200

```
N010 G91 G00 X0 Y0 Z10 (Linear interpolation)
N020 V.G.RW = 5 (Define a new distance to go)
N030 G108 (Activate edge banding)
```

(Linear interpolation)
(M97 is of mode MNE SNS,)
(Meas. run with lin. interpolation,)
(Continue with distance to go after probe signal)

```

\section*{Attention}

The M function for edge banding must be defined as an MNE_SNS mode (P-CHAN-00027).

\subsection*{4.2 Determining acceleration/deceleration (G08/G09/G900/G901)} Notice

Acceleration definition operations are only effective in connection with the linear slope.
\begin{tabular}{|l|l|l|}
\hline G08 & Acceleration at block start & (modal, initial state) \\
\hline G09 & Deceleration at block end & \begin{tabular}{l} 
(non-modal, effect dependent on G901/ \\
G900)
\end{tabular} \\
\hline G901 & \begin{tabular}{l} 
Deceleration after block end; can be cancelled blockwise by \\
G09
\end{tabular} & (modal) \\
\hline G900 & \begin{tabular}{l} 
Deceleration at block end independent of \\
G09 modal active
\end{tabular} & (modal) \\
\hline
\end{tabular}

If two consecutive NC blocks are programmed at different feedrates, a "soft" adaptation occurs at the block limit. As specified by G08, an acceleration takes place only at block start. G09 specifies that a deceleration to the feedrate of the next block should already occur at the end of the current block.

G09 is the default setting at program start. The G901 function defines the default setting for deceleration only after block end. The inverse G function G900 is a change-back function and is equivalent to the channel default setting.

\section*{Attention}

During transition from G00 to G01, G02 or G03, G09 is always active, i.e. deceleration at block end down to the velocity of the following block.
If the path velocity limit is already achieved while G901 is active, the velocity of the following block is reached at block end, i.e. then G901 is not effective.

\section*{Programing Example}

\section*{Define acceleration (G08/G09/G900/G901)}

Acceleration at block transition in initial state (equiv. to G08).
```

N10 G01 X500 F400
N20 X900 F1000

```


Fig. 31: Acceleration at block transition in the default state (corresp. to G08)

\section*{Programing Example}

\section*{Define acceleration (G900/G901)}

Deceleration at block transition with G901 and G900


Fig. 32: Deceleration at block transition with G901 and G900
\(\begin{array}{llll}\text { N10 G01 G900 X500 F1000 } \\ \text { N20 } & & \text { X900 F400 }\end{array}\)


Fig. 33: Deceleration at block transition with G901 and G900

\section*{Programing Example}

Define acceleration (G900/G901)
Combination of G09 with G901 and G900.
\begin{tabular}{lllll} 
N10 & G01 & G901 & X200 & F2000 \\
N20 & & G09 & X400 & F1600 \\
N30 & & X600 & F1200 \\
N40 & & G900 & X800 & F800 \\
N50 & & X1000 & F400
\end{tabular}


Fig. 34: Combination of G09 with G901 and G900

\section*{\(4.3 \quad\) Path/time-related feed interpolation (G193/G293)}

\section*{Notice}

Path-related feed interpolation is only effective in combination with linear slope (\#SLOPE [TYPE=STEP]) or HSC slope (\#SLOPE [TYPE=HSC]).
Time-related feed interpolation is only effective in combination with linear slope (\#SLOPE [TYPE=STEP]).
\begin{tabular}{|llc}
\hline G193 & Path-related feed interpolation & (non-modal) \\
G293 & Time-related feed interpolation & (non-modal)
\end{tabular}

When G193/G293 is selected, the feedrate between the initial and the programmed end velocity is linearly interpolated.

\section*{Notice}

It is not permitted to program G193 with G293, G08 or G09 in the same NC block.

\section*{Programing Example}

Path-related feed interpolation with (G193)
\begin{tabular}{llll} 
N10 & G01 & X500 & F1000 \\
N20 & G193 & X900 & F400 \\
N30 & X1000 & F400 &
\end{tabular}

At block transition N10/N20, path-related feed interpolation is activated and decelerated linearly to end velocity across the path programmed in N2O.


Fig. 35: Path-related feed interpolation with G193
Override changes are superimposed taking account of the permissible axis dynamics.

\section*{Programing Example}

Path-related feed interpolation with (G293)
\begin{tabular}{llll} 
N10 & G01 & X500 & F1000 \\
N20 & G293 & X900 & F400 \\
N30 & & X1000 & F400
\end{tabular}

Time-related feed interpolation is activated at block transition N10/N20 and linearly decelerated over time to end velocity.


Fig. 36: Time-related feed interpolation with G293

\section*{\(4.4 \quad\) Selection of planes (G17/G18/G19)}
\begin{tabular}{|lll|}
\hline G17 & X-Y plane & (modal, initial state) \\
G18 & Z-X plane & (modal) \\
G19 & Y-Z plane & (modal) \\
\hline
\end{tabular}

Programming G17, G18 or G19 defines the plane in which tool radius compensation and circular interpolation (see Section Circular interpolation (G02/G03) [ \(>55\) ] are to act:


Fig. 37: Display of plane selection (G17/G18/G19)

\section*{Tool radius compensation:}

Please note that tool radius compensation is always active on the first two main axes.

\section*{Circular interpolation:}

After a plane change with G17, G18 or G19, assignment again applies as per DIN: X, Y, Z are then assigned to I, J, K. The table below illustrates the syntax according to the selected interpolation plane:
\begin{tabular}{|l|l|l|l|}
\hline Plane & Interpolation type & Target point in plane & Centre point/radius \\
\hline G17 & G02/G03 & X..Y.. & I..J../R \\
\hline G18 & G02/G03 & Z..X.. & K..I../R \\
\hline G19 & G02/G03 & Y..Z.. & J..K../R \\
\hline
\end{tabular}

\subsection*{4.5 Mirroring in the plane (G21/G22/G23/G20)}

The term "virtual coordinates" is used to mirror in the plane. When mirroring is executed, the virtual coordinates ( \(\mathrm{x}_{\mathrm{m}}\) ) are the values entered in the NC program. By contrast, the real coordinates \(\left(\mathrm{x}_{\mathrm{r}}\right)\) are mirrored and are executed in reality. Mirror functions always act on the first and second main axes of the current plane ( \(\mathrm{G} 17, \mathrm{G} 18, \mathrm{G} 19\) ). The third main axis of the current plane is not mirrored.

The following mirroring functions are available (e.g. X-Y plane):
G21 Mirror programmed paths on the X axis (modal)
\[
\begin{aligned}
& x_{m}=-x_{r} \\
& y_{m}=y_{r}
\end{aligned}
\]

G22 Mirror programmed paths on the Y axis (modal)
\[
\begin{gathered}
x_{m}=x_{r} \\
y_{m}=-y_{r}
\end{gathered}
\]

G23 Superimpose G21 and G22 (modal)
\[
\begin{aligned}
& x_{m}=-x_{r} \\
& y_{m}=-y_{r}
\end{aligned}
\]

\section*{G20 Deselect the mirroring function (modal)}


Fig. 38: Virtual and mirrored (real) coordinates with G21
\begin{tabular}{ll}
\(x_{m}, y_{m}\) & Virtual coordinates \\
\(x_{r}, y_{r}\) & Mirrored coordinates \\
\(x_{a}, y_{a}\) & Absolute coordinates
\end{tabular}

Contours to be mirrored should be specified in a subroutine. This subroutine is then called after the mirroring function. Mirroring is modal

\section*{Notice}

If mirroring is programmed when tool radius compensation is selected, the side of the compensation also changes. This means that, when G41 is active, the equidistance to the programmed path after mirroring is calculated on the right of the contour (G42: TRC right of contour). This also happens even if the path direction is changed

Tool offsets and shifts (e.g. G54, G92, \#PSET...) are not mirrored with G21, G22 and G23. By contrast, the reference point offset (G92) is mirrored with the mirror function G351 [> 120].

\section*{Programing Example}

Mirroring in the plane (G21/G22/G23/G20)
```

%L DREIECK
N10 G90 X10 Y20
N20 G91 X10 Y-10
N30 X-10 Y-10
N40 Y20
N50 M29

```
\begin{tabular}{ll} 
\%SPIEGELUNG & \begin{tabular}{l} 
(Main program) \\
N10 G92 G90 X60 Y40 \\
N20 G01 F500
\end{tabular} \\
(Reference point offset) \\
(Straight line at feed 500) \\
N30 G21 & (Mirroring in the X axis) \\
N40 LL DREIECK & \begin{tabular}{l} 
(Call subroutine)
\end{tabular} \\
N60 G92 G90 X0 Y0 & (Reset the reference point offset) \\
N70 M30
\end{tabular}


Fig. 39: Example of mirroring

\section*{Attention}

\section*{Note the following when circles are mirrored:}

In general, only coordinates set in the NC program are mirrored (i.e. only the centre point coordinates for a full circle). After selecting the mirroring function, the motion path runs directly to the target point. The starting point of the motion path is not mirrored.
Consequence: In the motion block only the target point is mirrored but not the complete motion path (see Fig.: * Mirroring the target point in motion block")

Effect: If a full circle is programmed as a motion block and the starting/target points are not in 0/0, a new circle radius results from mirroring the centre point coordinates and the contour is changed (see Fig.: "Changing the contour when mirroring a full circle").

\section*{Effect:}


Fig. 40: Mirroring the target point in the motion block.


Fig. 41: Changing the contour when mirroring a full circle.

\section*{3D considerations:}

If the mirroring functions are selected in a different plane than the XY plane (G18, G19), note that the direction of rotation for circular interpolation may change depending on the selected plane: In addition, mirroring has no impact on the \(Z\) coordinate, i.e. the virtual and mirrored values in the \(Z\) direction are always identical.


If one views the ZX -Plane (view direction parallel to the Y -Axis), then the movements of the Modi G21 and G23 as well as G22 and G20 are one over the other.

If one views the YZ-Plane (view direction parallel to the X-Axis), then the movements of the Modi G21 and G20as well as G22 and G23 are one over the other.

Fig. 42: Effects of mirroring functions on the direction of circular rotation in different planes

\subsection*{4.6 Mirroring with axis specification (G351)}

G21 to G23 only select mirroring only for the first two main axes. The syntax with G351 described below permits free programming for axis mirroring.
G351 <Achsname> [ [+]|-] 1 \{ <Achsname> [ [+]|-] 1\(\}\) (non-modal)

Axis-specific selection of mirroring. The G function G351 is only valid as non-modal. However, mirroring is modal for an axis programmed with this function.
<Achsname> The axis coordinate defines whether to select or deselect mirroring in the axis. Coordinate value -1: Selects mirroring Coordinate value 1 or +1 : Deselect the mirroring function

Programing Example
Mirror with axis specification (G351)

G351 X-1 Y1 Z+1 (Select mirroring in the \(X\) axis and deselect mirroring)
(in the \(Y\) and \(Z\) axes)
- The axes mirrored can be programmed at any point in the NC block.
- At least one axis coordinate must be programmed together with G351.
- The G351 function must be programmed alone in the NC block. An exception is block number N.
- Repetitive selection or deselection of axis mirroring is permitted. However, an error message is output if repeated programming takes place in the same NC block.
- If mirroring is selected in synchronous mode for the lead axis (master axis), mirroring is not automatically selected for the tracking axis (slave axis). However, the slave axis is always tracked according to the path motions of the master axis. Therefore, an additional path motion resulting from mirroring the master axis always influences the slave axis.
- Mirroring is deselected for all axes at program start and reset. When axes are changed, mirroring of the changed axis is deselected.
- Mirroring the first or second main axis influences the path direction during circular interpolation and tool radius compensation.
- When mirroring is programmed when tool radius compensation is active, the selected side (G41/G42) is swapped automatically. This is only allowed for linear blocks.


Fig. 43: Mirroring the selected side with active tool radius compensation
- If a reference point offset G92 is active in an axis mirrored with G351, the coordinates of the reference point offset are also mirrored.


Fig. 44: Mirroring a reference point offset G92
- The coordinates of the circle centre point I, J, K are also mirrored (see Section Plane selection (G17/G18/G19) [> 114])
- When chamfers and roundings (G301/G302) are inserted, the I word is read as chamfer length or as radius. Therefore there is no need to consider mirroring here.

\section*{Programing Example}

\section*{Mirror with axis specification (G351)}

The examples below show how to use the G351 function. Assuming that the axes \(\mathrm{X}, \mathrm{Y}\) and Z are the 1 st, 2 nd and third main axis
\begin{tabular}{ll} 
N10 G351 X-1 & (Select mirroring in the \(X\) axis (G21)) \\
N20 G351 Y-1 & (Select mirroring in the \(Y\) axis (G22)) \\
N30 G351 X-1 Y-1 & (Select mirroring in the \(X\) and \(Y\) axis (G23)) \\
N40 G351 X1 Y+1 & (Deselect mirroring in the \(X\) and \(Y\) axes (G20)) \\
N50 X1 G351 Y-1 Z1 & \begin{tabular}{l} 
(Select mirroring in the \(Y\) axis and deselect) \\
(mirroring in the \(X\) and \(Z\) axis)
\end{tabular}
\end{tabular}
\begin{tabular}{|lcc|}
\hline 4.7 & Units \((\mathbf{G 7 0 / G 7 1 )}\) \\
\begin{tabular}{|llc|}
\hline G70 & Inputs in inch (inch) & (modal) \\
G71 & Inputs in metric units & (modal, initial state) \\
\hline
\end{tabular}
\end{tabular}

The statement G70 or G71 acts on all path and coordinate values.
Exception: Reading/writing tool parameters (V.G.WZxx.Pxx.) and kinematic parameters (V.G.WZxx.KIN_PARAMxx) always act directly regardless of the current unit.

\subsection*{4.8 Implicit subroutine calls (G80-G89/G800..)}
\begin{tabular}{|lll|}
\hline G80 - G89 [ [<Val1 \(>,<\) Val2 \(>,-,<\) Val50 \(>]]\) & Subroutine call & (non-modal) \\
or in addition & & \\
G800.. [ [<Val1>,<Val2>, - ,<Val50>]] & Subroutine call & (non-modal) \\
\hline
\end{tabular}

G80.. / G800.. When programming G80-G89, G800-G80-G89 and G800-G839**, an assigned global subroutine is implicitly called and executed. The default names of these subroutines can be configured either in the channel parameters P-CHAN-00160-P-CHAN-00169 and P-CHAN-00187 or defined during program runtime using the command \#FILE NAME [ 434].
If no program name is saved when G80-G89, G800-G819 and G800-G839** are programmed, the error message ID 20131 "Unknown G function" is generated. The global subroutine is called only once; this means G80-G89, G800-G819 and G800-G839** have no modal effect.
<Val1>, - ,<Val50> Optionally, a maximum of 50 transfer parameters (mathematical expressions in REAL format) can be bracketed in a fixed sequence to supply a subroutine (cycle). The parameters are separated by commas. Gaps in the sequence must be marked by consecutive commas ", ,".

By specifying transfer parameters, the subroutine call is handled as a cycle call according to the rules for cycles.
In analogy to cycle programming, the parameters can be read out in the subroutine using @Px accesses. There is a fixed assignment between the parameter and the @Px read access (e.g. @P1 reads parameter value 1, @P2 reads parameter value 2 and so on). Subroutines called in this way can also use the additionally extended cycle syntax with the @ character. The variable V.G.@P[i].VALID [> 584] in the subroutine (cycle) determines a parameter is programmed (valid).
**Extended to 40 calls (G800 - G839) as of V3.1.3079.23

A G80-G89 and G800.. is always executed as the last action at block end. This means that axis motions are programmed in the same NC block. They are always executed before the global subroutine is called.

\section*{Programing Example}

\section*{Implicit subroutine calls (G80-G89/G8xx)}

Assume the global subroutine g80_cycle.nc is called for G80:
```

N10 \#FILE NAME[ G80="g80_up_test.nc" ]
Nx ..
N30 G80 Call g80_up_test.nc as global subroutine
:
Assume the global subroutine g80_up_test.nc is called for G815:

```
```

N10 \#FILE NAME[ G815="g815_up_test.nc" ]
Nx ..
N30 G815 Call g815_up_test.nc as global subroutine
:
G85 calls the global subroutine cycle_test.nc with parameters:
N10 \#FILE NAME[ G85="cycle_test.nc" ]
Nx ..
N30 G85 [10,2, ,15,-3, ,5] Call cycle_test.nc as global subroutine

```

Example 2:
G803[5, @P1, @P2, @P3]
The meaning of the line above:
G803[@P1=5, @P2=@P1, @P3=@P2, @P4=@P3]
The result: all transferred parameters have the same value: 5

\subsection*{4.9 Dimension systems (absolute dimension/incremental dimension) (G90/G91)}
\begin{tabular}{ccc} 
G90 & Absolute dimension & (modal, initial state) \\
G91 & Incremental dimension & (modal)
\end{tabular}

With an absolute dimensional input (G90), all coordinate specifications are based on the coordinate origin, i.e. the coordinates in a motion block specify the target point in the coordinate system.
With incremental programming (G91), the coordinate values are based on the target point of the preceding motion block, i.e. the coordinates in a motion block specify the path to be travelled.

\subsection*{4.9.1 Exclusive programming}

In the default setting, only one measuring system may be selected in the NC block. It is not permitted to use redundant programming or programming of G90 and G91 in the same NC block. The position of G90/G91 within the NC block then has no meaning.

\section*{Programing Example}

\section*{Exclusive programming}
```

:
N10 X10 Y10 (Absolute measuring system G90 is selected, basic setting)
N20 G91 X20 Y20 (Deselect absolute and select relative programming)
N30 X30 G90 Y30 (Deselect relative and select absolute programming)
:
N100 G90 Z30 G90 (Error message: redundant programming of G90/G91)
N110 G91 X10 G90 Z30 (Error message: redundant programming of G90/G91)

```

\subsection*{4.9.2 Combined programming}

Channel parameter P-CHAN-00116 can be used to deselect the exclusive measuring system programming for path axis coordinates. Then it is possible to program absolute and relative measurements in the same NC block. It is also permitted to use the repeated programming of G90 and G91 in the same NC block.

The position of G90/G91 within the NC block then has a meaning. The measuring system last programmed is valid for all following path axis positions in the NC block and all other NC blocks up to the next G90/G91.

\section*{Programing Example}

\section*{Combined programming}
:
N10 X10 Y10 (Absolute dimensioning G90 is selected, default)
N20 G91 X20 G90 Y20 (relative for X axis, absolute for \(Y\) axis)
N30 X30 G91 Y30 Z20 (absolute for X axis, relative for Y/Z axis)
N30 G90 X30 G91 Y30 G90 Z20 (absolute for X/Z axis, relative for Y axis) .
N100 G90 G91 Z30 (relative for \(Z\) axis)
N110 G91 X10 G90 Z30 (relative for X axis, absolute for Z axis)

\section*{Notice}

G90/G91 has no influence on the auxiliary coordinates I, J, K of circular or helical interpolation. Its measuring system is defined exclusively by G161/G162.

\subsection*{4.10 \\ Exact stop (G60/G360/G359)}

\section*{G60}

Exact stop
(non-modal)
... or for exact stop across several blocks:
\begin{tabular}{llr} 
G360 & Select exact stop & (modal) \\
G359 & Deselect exact stop & (modal)
\end{tabular}

G60/G360 allow the exact approach to a target point within the exact stop limits. The feedrate is decreased to zero up to block end and decreases the following error.
The exact stop can be used if edges must be precision-machined or if the target point must be precisely reached in case of direction reversal. The motion is continued to the next position when a parameterisable control window is reached (P-AXIS-00236).

\subsection*{4.11 Polynomial contouring (G61/G261/G260)}
\begin{tabular}{|lcc|}
\hline G61 & Polynomial contouring (at block end) & (non-modal) \\
\(\ldots .\). or for polynomial contouring across several blocks: & \\
G261 & Select polynomial contouring (at block end) & (modal) \\
G260 & Deselect polynomial contouring & (modal) \\
\hline
\end{tabular}

Polynomial contouring is the constant curvature and constant direction connection between two motion blocks. For this purpose, the originally programmed motion blocks are shortened. A contouring curve is added between the blocks. This process permits contouring between the transitions straight line - straight line, straight line - circle and circle - circle (see Figure below). It is not restricted to a particular plane but allows contouring between any number of curves located in space.

Motion blocks without motion path are not considered here. (See Relevant block length [> 262] ).


Fig. 45: Examples of polynomial contouring

\section*{The following contouring types are available:}
- Contouring with corner deviation
- Dynamic optimised contouring
- Dynamic optimised contouring with master axis
- Contour with interim point
- Dynamically optimised contouring of the contour.

Depending on the contouring type, different conditions may be specified. The parameters remain valid until the contouring process is fully executed. If the contouring parameters between the preblock and post-block are changed, the change becomes effective with the next contouring operation.

For further information see Section Polynomial contouring for long blocks (G61/G261/G260) [ 258].

\subsection*{4.12 \\ Corner deceleration}

In the milling process, the removal volume \(V_{z}\) and the associated spindle performance required in the inside corners increases as a function of block transition geometry since the tool already removes material from the subsequent contour (see figure below).
If machining is always performed at the spindle's performance limit, the feedrate in the corner section must be reduced so that there is also sufficient spindle performance in the inside corners. In order to stay within the spindle's performance limits, it must be possible in the NC program to define a point on the path after which the velocity can be reduced. There are three NC commands available for this corner deceleration: one for parameterisation, one for activation and one for deactivation.


Fig. 46: Changing the removal volume Vz over time at an inside corner of \(90^{\circ}\) and at constant feedrate

\subsection*{4.12.1 Parameterising corner deceleration (\#CORNER PARAM)}

\section*{Release Note}

As of Build V2.11.2010.02 replaces the command \#CORNER PARAM [...] the command \#SET CORNER PARAM [...]assigns the definition of the main spindle. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.
\#CORNER PARAM [ DIST<expr> UNIT<ident> FEED<expr> ]

DIST<expr>
UNIT<ident>

FEED<expr>

Corner distance in [mm, inch]
Corner feed rate unit FEED<expr>. Permitted identifiers:
FWORD Unit according to current F word
PERCENT Unit in [\%]
Corner feed rate, [according to UNIT<ident>]

The point in space after which the path feedrate is to be reduced linearly along the path can be calculated by specifying a corner distance. The corner distance to be programmed refers here to the corrected path, not to the originally programmed target points.

\section*{Notice}

Corner deceleration is only effective in combination with the linear slope P-CHAN-00071.


Fig. 47: Representation of feed at a circular inside contour

\subsection*{4.12.2 Selecting/deselecting corner deceleration (G12/G13)}
\begin{tabular}{|lll|}
\hline G12 & Deselecting corner deceleration & (modal, initial state) \\
G13 & Selecting corner deceleration & (modal) \\
\hline
\end{tabular}

An error message is output if G13 is programmed without previously defining the corner deceleration parameter.
An error message is output if G12 and G13 are programmed at the same time in an NC block.
These G functions are modal. The initial state after system start-up is "Corner deceleration inactive" (G12).
No error message is output if corner deceleration (G13) and end of program (M30) are selected. The initial position G12 is selected before the program is started.
An error message is output at a corner distance of 0 .

\section*{Programing Example}

\section*{Selecting and deselecting corner deceleration (G12/G13)}
```

N010 G01 G90 X10 Y10 F1000
N015 \#CORNER PARAM[DIST=10 UNIT=PERCENT (Corner distance 10mm)
FEED=50]
(Corner feed 50%)
N020 G13
(Activation)
N030 X50 (Corner deceleration active)
N040 Y50 (Corner deceleration active)
N050 G12 (Deselection)
N060 X30 Y30
N050 M30

```

Please note that corner deceleration is not activated until block N030. The block transition N010/ N030 is not affected by corner deceleration.
Similarly, corner deceleration is still active in block N040 because it was not deselected.
Deselecting corner deceleration must be programmed before or in the NC block in which corner deceleration should not longer be active.

\subsection*{4.13 Zero offsets (G53/G54/...G59)}

In this section zero offset is abbreviated to 'NPV'.
\begin{tabular}{|l|l|l|}
\hline G53 & Deselect zero offset * & (modal, initial state) \(^{*}\) \\
\hline G54 & (Activate CS[1]) NPV & (modal) \\
\hline G55 & (Activate CS[2]) NPV & (modal) \\
\hline G56 & (Activate CS[3]) NPV & (modal) \\
\hline G57 & (Activate CS[4]) NPV & (modal) \\
\hline G58 & (Activate CS[5]) NPV & (modal) \\
\hline G59 & (Activate CS[6]) NPV & (modal) \\
\hline
\end{tabular}

The significance of the G53 data record can be controlled by P-ZERO-00001. Depending on parameterisation, G53 means either that NPV is deselected or the G53 data record is used as an addition NPV.

The default setting of the NPV which is automatically active in initial state is also parameterisable P-ZERO-00002.

G54... G59 provide the corresponding zero offsets from the zero offsets table. The zero offset is already valid in the block in which G53, G54, etc. is programmed. However, no path motion can take place without specifying any coordinates.
In initial state (G53) no zero offsets are active.

\section*{Notice}

In addition, the following applies when selecting a zero offset in G91 mode:
The programming of...
N10 G54
N20 G0 X0 G91
... Does not call motion on the \(X\) axis since this is a motion relative to zero.
As a result, a zero offset is only effective when the next motion datum is programmed in absolute coordinates (G90).

\subsection*{4.13.1 Enhanced zero offset variables}
\begin{tabular}{|l|l|l|l|}
\hline Variable & Meaning & Read & Write \\
\hline V.G.NP[j].ALL & Address all axes of a zero offset & yes & yes \\
\hline V.G.NP_AKT.V[i] & Current (currently active) zero offset of an axis & yes & yes \\
\hline V.G.NP_AKT.ALL & Address currently active zero offsets of all axes & yes & yes \\
\hline V.G.NP_DEFAULT & Index of effective zero offset in initial state & yes & yes \\
\hline
\end{tabular}

Write access via the V.G.NP[j] and V.G.NP_DEFAULT variables remains effective until the next list interpretation (explicit request or start-up). By contrast, write accesses via V.G.NP_AKT are only effective on zero offsets currently stored in the decoder. When the NPV previously written via V.G.NP_AKT is re-selected, the original values from the list again become effective. The current data record must be saved to the list by assignment via V.G.NP[j].ALL to retain write access to V.G.NP_AKT after deselection.

\section*{Programing Example}

\section*{Enhanced zero offset variables}

Saving the currently active zero offset values:

\subsection*{4.13.2 Adding and subtracting offsets}

The V.G.NP[j].ALL variable re-assigns complete NPV data records by the additive assignment of existing NPVs. The offsets of all axes are included in the various calculations. The notations `+=`, \(-=`\), and `=` are permitted in NPV assignment.

\section*{Programing Example}

G54 (NPV1) is assigned the combination of G55 (NPV2) and G57 (NPV4):

N10 V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL

\section*{Programing Example}

The same operation but with the inclusion of G54:

N10 V.G.NP[1].ALL = V.G.NP[1].ALL + V.G.NP[2].ALL + V.G.NP[4].ALL
or
N10 V.G.NP[1].ALL += V.G.NP[2].ALL + V.G.NP[4].ALL

\section*{Attention}

It is not possible to interconnect V.G.NP[j].ALL variables with axis-specific V.G.NP[j].V[i] variables or constants within an assignment.

\section*{Programing Example}

G54 (NPV1) is defined by the combination of G55 (NPV2), the X offset of G57 (NPV4) and a compensation value:

\section*{WRONG:}

N10 V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].V.X + 100
RIGHT:
A assignment must be made in two steps:
```

N10 V.G.NP[1].ALL = V.G.NP[2].ALL
N20 V.G.NP[1].V.X = V.G.NP[4].V.X + 100

```

\subsection*{4.13.3 Access to the current zero offset}

The currently active NPV in the decoder is accessed via the V.G.NP_AKT.V[i] variable. The operator need not know which NPV (i.e. which index) is currently selected.

\section*{Programing Example}

The current NPV of the X axis should be 200 .

N10 V.G.NP_AKT.V.X = 200
The current NPV in all axes should be expanded by the offset values from G55 (NPV2).
N10 V.G.NP_AKT.ALL = V.G.NP_AKT.ALL + V.G.NP[2].ALL
or
N10 V.G.NP_AKT.ALL += V.G.NP[2].ALL
Changing the current NPV is not effective across programs. However, by adding an assignment, the operator can can also use the current NPV in other programs.

\section*{Programing Example}

The current NPV should be saved under G54 (NPV1) for future utilisation in other programs.
```

N10 V.G.NP[1].ALL = V.G.NP_AKT.ALL

```

\subsection*{4.13.4 Default zero offset}

The active NPV after program start is parameterisable (P-ZERO-00002).
In the NC program, the operator can use the V.G.NP_DEFAULT variable to access the default index. The change is valid across all programs.

\section*{Programing Example}

Starting with this program line, G55 (NPV2) will be the new default NPV. This means that G55 is automatically active at next program start.

N100 V.G.NP_DEFAULT \(=2\)

\subsection*{4.13.5 Creating zero offset groups}

A zero offset group (NPVG) defines the rules to create an NPV, i.e. it describes the components contained in a specific NPV data record. An NPVG is defined in the NC program or by pre-assignment in the channel parameter list [1] [ \(>819]-1\) in conjunction with a symbolic string. The use of symbolic strings is described in the Section Macros [ 709]. Three steps are basically required to activate an NPVG:

\section*{Programing Example}

Creating zero offset groups

\section*{1st Step:}

Define an NPVG in the NC program, e.g. under the macro name VERSCH_1:
```

N10 " VERSCH_1 " = " V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL "

```

Or in the channel parameter list [1] [ \(>819]-1\) :
```

makro_def[0].symbol VERSCH_1
makro_def[0].nc_code V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL

```

\section*{2nd Step:}

Creating the NPV (new assignment of the NPV1 data record):
```

N20 " VERSCH_1 "

```

\section*{3rd Step:}

Select the NPV (here: NPV1, i.e. G54)
N30 G54

\subsection*{4.13.6 \\ Extended zero offset (G159)}

G159 = <index>
(modal)

G159 Extended zero offset
<index>
List index of the zero data record

Access to additional NPV data records [3] [ 819]. It is also possible to access the G53...G58 data records. The maximum number of extended NPVs is parameterisable [6] [ 819]-6.12.


\section*{Programing Example}
```

G159 = 1 ; corresponds to G54
G159 = 2 ;corresponds to G55
G159 = 3 ; corresponds to G56
G159 = 4 ; corresponds to G57
G159 = 5 ; corresponds to G58
G159 = 6 ;corresponds to G59
G159 = 7 Data record with list index 7
:
G159 = 10 ; Data record with list index 10

```
G159 = 0 ; corresponds to G53

\subsection*{4.13.7 Enable/disable zero offsets axis-specific (G160)}
```

G160 = <index> <axis_name><flag>

| G160 | Axis-related zero offset |
| :--- | :--- |
| <index> | List index of the zero data record |
| <axis_name><flag> | Axis with validity flag of its zero offset |


| Flag | Meaning |
| :--- | :--- |
| 0 | Zero offset of the axis is included in the calculation. |
| 1 | Zero offset of the axis is not included in the calculation. |

You can define in each zero offset data record <index> previously by setting of the parameters P-ZERO-00004 the axes <axis_name> for which the offset is or is not to be included in the calculation. This means, individual axis offsets can be disabled or enabled.

This axis-specific validity of a zero offset can be changed in the NC program using G160. Programing Example
Before selecting G55 (zero offset data record with index 2 ) the offsets of the $X$ and $Z$ axes are disabled and the offsets of the Y axis are enabled.

```
N10 G160 = 2 X1 Y0 Z1
N20 G55
:
```

In the next motion block, therefore, only the axis offsets of G55 are included in the motion if they are not disabled ( Y axis).

### 4.14 Specifying centre point for circle definition (G161/G162)

| G161 | Circle centre point absolute | (modal) |
| :--- | :--- | :--- |
| G162 | Circle centre point relative | (modal, initial state) |

When G162 is active (initial state), the centre point is defined by $\mathrm{I}, \mathrm{J}$ and K relative to the circle starting point.
When G161 is active, I, J and K are specified absolute in the programmer's coordinate system.

## $4.15 \quad$ Radius programming (R/G163)

Radius programming is expected if no circle centre point (with I, J and/or K) is specified. The value following the character $R$ is interpreted as the radius in [mm, inch]. It is automatically used in lines with circular programming. As a result, the radius statement is modal and, in case of multiple circular path motions with the same radius, it is unnecessary to repeat the radius specification.

## Notice

The maximum permissible circle radius is $10^{9} \mathrm{~mm}$. However, the target point of the arc may not exceed the maximum permissible motion path of the axes of $+-2,14^{*} 10^{5} \mathrm{~mm}$.

## Notice

If the radius value is possible, the shortest possible arc is defined; if the radius value is negative, the largest possible circle is defined (see figure below).


With indexed radius programming, the radius can be specified as "R1". In this case, the index may only have the value 1. If "R1" is used on the right-hand assignment side, the index value 1 may not be programmed as a mathematical expression.

## Programing Example

Radius programming (R-, G163)

```
1.)
N10 P2 = 1
N20 RP2 = 5 ;permitted
2.)
N10 R1 = 5
N20 P2 = R1 ;permitted
3.)
N10 R1 = 5 P2 = 1
N20 P3 = RP2 ; not permitted
```


## Programing Example

These examples produce semicircles of radius 50.
1.)

N10 G90 G01 X0 Y0 F500
N20 G02 X100 R50 ; clockwise semicircle
N30 G03 X200 R50 ; counter-clockwise semicircle
2.)

N10 G90 G01 X0 Y0 F500
N20 G02 R=50 ; no motion as yet here
N30 X100 ;clockwise semicircle
3.)

N10 G90 G01 X0 Y0 F500
N20 R1=50
N30 G02 X100 ; clockwise semicircle
N40 G03 X200 ; counter-clockwise semicircle
4.)

N10 G90 G01 X0 Y0 F500
N20 G02 X100 R1=50 ; clockwise semicircle
N30 G03 X200 ; counter-clockwise semicircle

The programming below results in an error message since $R 1$ is interpreted as a radius of value 1.
N10 G90 G01 X0 Y0 F500
N20 R1=50
N30 G02 X100 R1

As an alternative to circle definition with R or R 1 , the circle radius can be specified using the following G code:

G163 $=$ <Radius $>$ (modal)

G163
<Radius>

Circle radius specification
Circle radius value in [mm, inch]

The circle definition using radius is valid when circular interpolation is selected until it is redefined or until it is deselected by specifying an I and/or J and/or K.

## Programing Example

Circle radius programming with G163
(N10: Motion to origin)
(N20: Clockwise semicircle with target value X100 and under preset)
(of the circle radius by G163 (radius specified is modal))
(N30: Counter-clockwise semicircle with target value X200 and radius,)
(which was defined in N 20 and is modal)
\%Radiusprogramming_G163
N10 G90 G01 X0 Y0 F1000
N20 G02 G163=50 X100 ; clockwise semicircle
N30 G03 X200 ; counter-clockwise semicircle
N40 M30


## Attention

If the starting and end points of the circle programmed with "R", "R1" or "G163" are identical, an error message is output. If a full circle is travelled, it must be programmed with $\mathrm{I} / \mathrm{J} / \mathrm{K}$.

### 4.16 Controlling centre point offset in circle (G164/G165)

| G164 | Centre point compensation OFF | (modal) |
| :--- | :--- | :--- |
| G165 | Centre point compensation ON | (modal, initial state) |

When G165 is active, a circle programmed by an $I$, $J$ and $K$ statement is compensated in such a way that an arc can be interpolated if the circular direction (G02/G03), start position (end point of the previous block) and end point (coordinates in the circular block). The centre point programmed with I, J and K may then be offset. The more precise the centre point is specified, the less the centre point offset will be.

## Notice

If the circle is programmed by specifying a radius $R$, no circle centre point compensation is effective since the centre point is then always calculated exactly here.

For the deviation from programmed to compensated centre point, two limit values P-CHAN-00059 and P-CHAN-00060 are monitored. If they are exceeded, an error message is output:
mittelpkt_diff:
Permissible deviation in $10-4 \mathrm{~mm}$
mittelpkt_faktor:
Percentage deviation in 0.1\%

A check is made whether the centre point offset $\Delta \mathrm{m}$ is greater than the absolute value "mittelpkt_diff"

$$
\Delta \mathbf{m}>\text { mittelpkt_diff? }
$$

and whether the centre point offset $\Delta m$ is greater than the product of "mittelpkt_faktor/1000" and the corrected radius "radius".

$$
\Delta \mathrm{m}>\text { mittelpkt_faktor/1000 * radius? }
$$

Therefore, the upper limit for $\Delta \mathrm{m}$ is linearly dependent on the calculated radius. This results in the relationship contained in the figure below between centre point offset $\Delta \mathrm{m}$ and the calculated radius "radius".

## Example:

"mittelpkt_faktor" = 5 signifies that the distance between the programmed centre point coordinates and the compensated centre point coordinates may be maximum $0.5 \%$ of the compensated radius of the circle.


Fig. 48: Relationship between centre point offset Dm and the calculated radius "radius"
The programmed centre point coordinates must then lie in a circumcircle about the compensated centre point of the circle. The radius of this circumcircle corresponds to the permissible centre point offset $\Delta \mathrm{m}$ which can be set using the two parameters 'mittelpkt_diff' and 'mittelpkt_faktor':


Fig. 49: Area of permissible programmed centre points

### 4.16.1 Special function: circle radius compensation in combination with G164

In certain situations, circle centre point compensation (G165) can lead to an unfavourable offset in the programmed circle centre point and therefore the circle's position. Such unfavourable situations may occur when the circle starting and target points are close together and the circle is almost equivalent to a programmed full circle.

This type of circle with specified target point with and without circle centre point compensation ( $\mathrm{G} 165 / \mathrm{G} 164$ ) is programmed in the example below. To simulate the resolution error in the postprocessor and when circle centre point compensation is active, the circle's target point is shifted by $0.1 \mu \mathrm{~m}$ in each case in the x and y directions relative to the starting point. The circle with G 165 is rotated about the starting point and the position of the compensated circle centre point $M_{K}$ shifts very considerably relative to the programmed centre point M .

## Programing Example

```
N10 G00 G90 X0 Y0 Z0
N20 G01 X-50 Y0 F20000
N30 G01 X0
N40 G165 G02 X0.0001 Y0.0001 J200
N50 G01 Y50
M30
N10 G00 X0 Y0 Z0
N20 G01 X-50 Y0 F20000
N30 G01 X0
N40 G164 G02 X0.0 Y0.0 J200
N50 G01 Y50
N60 M30
```



Fig. 50: Circle centre point shift in the case of G165
In such cases a better result can generally be obtained when circle centre point compensation (G164) is disabled and when max_radius_diff_circle > 0 ( $\mathrm{P}-\mathrm{CHAN}-00171$ ) and max_proz_radius_diff_circle (P-CHAN-00172) are set. The - programmed circle centre point is not changed by the function and the circle radius difference is transferred linearly by the circle's angle from the starting radius to the target radius.
If the circle radius deviations lie in the order of magnitude of the resolution accuracy, the circle distortions and dynamic effects are generally negligible.

## Notice

For a full circle, the circle starting and target points must be identical in this case.

## $4.17 \quad$ Feedforward control (G135/G136/G137)

| G135 | Select feedforward control | (modal) |
| :--- | :--- | :--- | :--- |
| G136 <Achsname><expr> \{ <Achsname><expr> \} | Specify weighting | (modal) |
| G137 | Deselect feedforward control | (modal) |

Path distortions can be reduced by using velocity and acceleration feedforward control.
Axis group.specific activation is programmed with G135. An axis-specific percentage weighting of the calculated feedforward control variables in [\%] takes place with G136. It is limited to $100 \%$ for all axes.

G137 deactivates axis group-specific feedforward control. In case of axes for which no feedforward control is to be implemented after global selection with G135, a percentage weighting of 0\% must be specified with G136. It is also possible to enter the selection and weighting of feedforward control in a single block.

At every program start, feedforward control is explicitly disabled in the interpolator and the weighting factors are set to $100 \%$.
If feedforward control is disabled or enabled during the NC program, the weighting factors remain at the values set by G136 or, if no G136 is programmed, to $100 \%$.

## Attention

After an axis exchange, the G136 weighting factors are reset to 100\% for all axes involved.

## Programing Example

Feedforward control (G135/G136/G137)

```
G135 (Select feedforward control: weighting for all axes 100%)
G136 X80 Y95 Z0(Weighting; Z axis has no feedforward control here)
G137 (Deselect feedforward control: weighting for all axes 100%)
```


### 4.18 Weighting of maximum velocity (G127/ G128)

```
G127 <axis_name><expr> { <axis_name><expr> } modal
G128 = <expr> modal
```

| G127 | Axis specific weighting of maximum velocity |
| :--- | :--- |
| <axis_name><expr> | Weighting for specific axes in [\%] |
| G128 | Axis group-specific weighting of maximum velocity <br> <expr> |
| weighting for all axes in [\%] |  |

Using the G127/G128 function it is possible to change the maximum velocity.
The maximum velocity can be influenced by a percentage change in the associated velocity characteristic value.

If programming takes place with G127/G128, all axes which are not programmed or not yet programmed are set to $100 \%$. Each further selection of this function, irrespective of the previous programs, signifies $100 \%$. That means, the geometrical data processing always weights the default value P-AXIS-00212 with the percentage value.

Therefore, $50 \%$ programmed twice in succession means the setting is made to $50 \%$ and not to 25\%.

## Notice

Feed weighting at maximum velocity does not act on single axis motions e.g. homing, manual mode or independent axes.

Attention
If $\mathrm{G} 127 / \mathrm{G} 128>100 \%$ the maximum velocity is limited to the MAX value P-AXIS-00212.
If $\mathrm{G} 127 / \mathrm{G} 128=0$, the maximum velocity is limited to the minimum value $1 \mu \mathrm{~m} / \mathrm{s}$.

## Programing Example

## Weighting of maximum velocity (G127/ G128)

```
N10 G127 X70 Y60
;Axis-specific weighting of vb_max
; vb max of X axis is restricted to 70%
; vb_max of Y axis is restricted to 60%
N20 G128 = 100 ;Axis group-specific weighting of vb_max
;vb_max of all axes to 100%
Special feature:
N20 G128 = 100 X10 Y20
;Using G128, axis positions
; can also be programmed within the same block!
```


## $4.19 \quad$ Weighting of rapid traverse velocity (G129)

```
G129 = <expr>
(modal)
```

G129 Axis group-specific weighting of rapid traverse
<expr>
Weighting for all axes in [\%]

The G129 functions can be used to change the rapid traverse velocity G00.
The feedrate can be influenced by a percentage change in the associated velocity characteristic value.

If programming takes place with G129, all axes which are not programmed or not yet programmed are set to $100 \%$. Each further selection of this function, irrespective of the previous programs, signifies 100\%. That means, the geometrical data processing always weights the default values P-AXIS-00209 with the percentage value.
Therefore, $50 \%$ programmed twice in succession means the setting is made to $50 \%$ and not to 25\%.

## Notice

Feed weighting only acts on rapid traverse NC blocks (G00). It has no effect on single axis motions e.g. homing, manual mode or independent axes.

## Attention

If G129 is $\mathbf{> 1 0 0 \%}$, rapid traverse velocity is limited to the MAX value P-AXIS-00212.
If $\mathrm{G} 129=0$, rapid traverse velocity is limited to the minimum value $1 \mu \mathrm{~m} / \mathrm{s}$.

Programing Example
Weighting of rapid traverse velocity (G129)

```
N10 G129 = 70
(Axis group-specific rapid traverse weighting)
(Rapid traverse velocity of all axes to 70%)
N20 G00 X100 Y150
(Linear interpolation, rapid traverse motion with 70%)
Special feature:
N50 G129 = 70 X10 Y20
(With G129 in the same block, axis positions)
(can also be programmed)
```


### 4.20 Parameterising the acceleration profile

4.20.1 Acceleration weighting (G130/G131/G231/G333/G334)

| Weightings for accelerations: |  |
| :--- | :--- |
| G130 <Achsname>=.. \{<Achsname>=..\} | (modal) |
| G131 $=.$. | (modal) |
| G231 $=.$. | (modal) |
| Weightings for accelerations in feedhold: |  |
| G333 <Achsname>=.. \{ <Achsname>=..\} | (modal) |
| G334=.. |  |

G130 <Achsname>=<ex- Axis-specific acceleration weighting, weighting for specific axes in [\%], only acts on feed blocks G1/G2/G3.
G131=<expr> Axis group-specific acceleration weighting with G01, G02, G03, weighting for all axes in [\%]
G231=<expr> Axis group-specific acceleration weighting with G00, weighting for all axes in [\%]
G333 <Achsname>=<ex- Axis-specific acceleration weighting with feedhold, weighting for specific axes in [\%]
pr>
G334=<expr> Axis group-specific acceleration weighting with feedhold, weighting for all axes in [\%]

The G130/G131/G231 functions can change acceleration ramps.
This acceleration can be influenced by a percentage change in the corresponding default assignment of acceleration characteristic values. With a jerk-limited profile, these values are the axis parameters P-AXIS-00001 and P-AXIS-00002.

When P-CHAN-00097 is set, the feedhold parameter list is used for the deceleration process. In this case, the deceleration of the feedhold ramp P-AXIS-00024 by a percentage change using G333/G334. Feedhold weighting is only effective if the resulting deceleration is equal to or greater than the active values of the weighted G01/G00 accelerations.
When the function is programmed with G130/G131/G231/G333/G334, all axes which are not programmed or not yet programmed are set to $100 \%$. Every additional selection of these functions, irrespective of previous programming, refers to $100 \%$. This means that geometrical data processing always weights the default values [2] [ $>819]-1$ and/or [2] [> 819]-2 with the percentage value.

Therefore, $50 \%$ programmed twice in succession means the setting is made to $50 \%$ and not to $25 \%$. A weighting of over $100 \%$ is possible up to maximum axis acceleration P-AXIS-00008.
Alternatively, a reduction in acceleration can take place using the "Reducing path acceleration" control unit.

## Attention

After an axis exchange, the G130/G333 weighting factors are reset for all axes involved to $100 \%$.

## Notice

Acceleration weighting does not act on single axis motions, e.g. homing, manual mode or independent axes.

## Programing Example

Acceleration weighting (G130/G131/G231/G333/G334)

| N10 G130 X70 | ;Axis-specific weighting of acceleration <br> ;Acceleration of X axis is restricted to 70\% |
| :---: | :---: |
| N20 G01 F1000 X100 | ; Linear interpolation |
| N30 G130 Y60 | ;Acceleration of $Y$ axis is restricted to 60\% <br> ;Acceleration of X axis remains at $70 \%$ |
| N40 Y100 | ; Linear interpolation |
| N50 G131 $=100$ | ;Axis group-specific acceleration weighting <br> ; G01,G02,G03 acceleration of all axes to 100\% |
| N60 G231 $=80$ | ;Axis group-specific acceleration weighting <br> ;G0 acceleration of all axes to 80\% |
| N70 G00 X200 | ; Rapid traverse |
| N80 G333 X150 | ```;Axis-specific weighting of acceleration ;with feedhold. Deceleration of X axis is increased to ; 150%.``` |
| N90 G334 $=200$ | ```;Axis group-specific acceleration weighting ;with feedhold. Deceleration of all axes is increased to ; 200%.``` |
| Special feature: |  |
| N50 G131 = $100 \mathrm{X10}$ Y20 | ; With G131/G231/G334 axis positions can ;also be programmed in the same block! |

### 4.20.2 Ramp time weighting (G132/G133/G134/G233/G338/G339)

| Weightings for ramp times: |  |
| :--- | :--- |
| G132 <axis_name>=.. \{ <axis_name>=.\} | (modal) |
| G133 $=.$. | (modal) |
| G134 $=.$. | (modal) |
| G233 $=.$. | (modal) |
| Weightings for ramp times in feedhold: | (modal) |
| G338 <axis_name>=.. \{ <axis_name>=.\} | (modal) |
| G339 $=.$. |  |

G132 <axis_name><ex-
pr>
pr>
G339=<expr>

G133=<expr> Axis group-specific ramp time weighting with G01, G02, G03, weighting for all axes in [\%]
G134=<expr> Axis group-specific weighting of geometrical ramp time, weighting for all axes in [\%]
G233=<expr> Axis group-specific ramp time weighting with G00, weighting for all axes in [\%]
G338 <Achsname>=<ex- Axis-specific ramp time weighting with feedhold, weighting for specific axes in [\%]
Axis-specific ramp time weighting, weighting for specific axes in [\%], only acts on feed blocks G1/G2/G3.

Axis group-specific ramp time weighting with feedhold, weighting for all axes in [\%]

These ramp times can be influenced by a percentage change in the corresponding default ramp times. The G132/G133/G233 functions can change the ramp time of axis acceleration with a nonlinear slope [2] [ 819]-1. (If the slope is linear, the acceleration curve is step-shaped (see figure in Section Jerk-limiting slope [> 371])

The G134 function can change the geometrical ramp time of a non-linear slope (P-AXIS-00199).
When P-CHAN-00097 is set, the feedhold parameter list is used for the deceleration process. In this case, the deceleration of the feedhold ramp P-AXIS-00081 by a percentage change using G338/G339.

When the function is programmed with G132/G133/G134/G233/G338/G339, all axes which are not programmed or not yet programmed are set to $100 \%$. Every additional selection of these functions, irrespective of previous programming, refers to 100\%. This means that geometrical data processing always weights default values with the percentage value. Therefore, $50 \%$ programmed twice in succession means the setting is made to $50 \%$ and not to $25 \%$.

## Attention

After an axis exchange, the G132/G333 weighting factors are reset for all axes involved to $100 \%$.

## Notice

Ramp time weighting does not act on single axis motions e.g. homing, manual more an independent axes.

## Programing Example

Ramp time weighting (G132/G133/G134/G233/G338/G339)

| N10 G132 X200 | ;Axis-specific weighting of ramp time <br> ; Ramp time of X axis is increased by 200\% |
| :---: | :---: |
| N20 G01 F1000 X100 | ; Linear interpolation |
| N30 G132 Y50 | ; Ramp time of $Y$ axis is decreased by 50\% <br> ; Ramp time of X axis remains at 200\% |
| N40 Y100 | ; Linear interpolation |
| N50 G133 $=100$ | ;Axis group-specific weighting of ramp time ;G01,G02,G03 ramp times of all axes to 100\% |
| N60 G134 $=50$ | ;Axis group-specific weighting of ramp time <br> ; Geometrical ramp time of all axes to $50 \%$ |
| N70 G233 $=80$ | ;Axis group-specific weighting of ramp time ; GOO ramp times of all axes to 80\% |
| N80 G00 X200 | ; Rapid traverse |
| N90 G338 X150 | ```;Axis-specific weighting of ramp time ;with feedhold. Ramp time of X axis is increased to ; 150%.``` |
| N100 G339 $=200$ | ; Axis group-specific weighting of ramp time <br> ; with feedhold. Ramp time of all axes is increased to ; 200\%. |
| Special feature: |  |
| N50 G133 = $100 \mathrm{X10}$ Y20 | ;With G133/ G134/G233/G339 axis positions can ;also be programmed in the same block! |



Fig. 51: Example of ramp time weighting with G132/G133/G233

The figure below shows the influence of G134 on acceleration perpendicular to the path (centrifugal acceleration a1--> a2) with increasing feed (v1 --> v2) during circular motion (P1 --> P2).
If the change in centrifugal acceleration is reduced by increasing the G 134 ramp time, acceleration is smoother and target acceleration a2 is not reached until point P3.


Fig. 52: Ramp time weighting with G 134 and with circular interpolation

### 4.21 Machining time or feedrate (G93/G94/G95/G194)

| G93 | Specify machining time in seconds | (modal) |
| :--- | :--- | :--- |
| G94 | Feed per minute | (modal, initial state) |
| G95 | Feedrate per revolution | (modal) |
| G194 | Feedrate calculation based on maximum weighted axis feeds | (modal) |

The action of the F word can be optionally switched over by using the G functions G93, G94, G95 and G194.
G93 combined with the $F$ word defines a machining time in [s].
G94 combined with the F word:

- feed in [ $\mathrm{mm} / \mathrm{min}, \mathrm{m} / \mathrm{min}$, inch/min] for linear axes
- feed in [ ${ }^{\circ} / \mathrm{min}$ ] for rotary axes

G95 combined with the $F$ word defines a feedrate per revolution in [ $\mathrm{mm} / \mathrm{U}$, inch/U]. The function is described in greater detail in Section Feedrate per revolution (G95) [> 637].
G194 combined with the F word defines a weighting factor in [\%] for maximum feedrates. The maximum permissible feedrate on the path then results from the axis-specific values P -AXIS-00212. At least one axis then moves at its maximum weighted velocity. Only weighting factors less than $100 \%$ are permitted.

## Programing Example

Machining time or feedrate (G93/G94/G95/G194)

```
```

N10 G90 F1000 X100 (Feedrate 1000 mm/min (G94 default))

```
```

N10 G90 F1000 X100 (Feedrate 1000 mm/min (G94 default))
N20 G194 F90 (Weighting 90% to max. axis feedrate)
N20 G194 F90 (Weighting 90% to max. axis feedrate)
Nxx X200 (Feed e.g. 9000 mm/min at vb_max=10000 mm/min)
Nxx X200 (Feed e.g. 9000 mm/min at vb_max=10000 mm/min)
N80 G94 X50
N80 G94 X50
(Feedrate 1000 mm/min valid from N10)
(Feedrate 1000 mm/min valid from N10)
Nxx X.. Y.. Z.. (Interpolation)
Nxx X.. Y.. Z.. (Interpolation)
N120 G93 F20 (Machining time 20 s]
N120 G93 F20 (Machining time 20 s]
Nxx X.. Y.. Z..
Nxx X.. Y.. Z..
N160 G94 F1500 X150
N160 G94 F1500 X150
Nxx X.. Y.. Z..
Nxx X.. Y.. Z..
N200 M30

```
N200 M30
```

```
N10 G90 (Feedrate 1000 mm/min (G9 default))
```

N10 G90 (Feedrate 1000 mm/min (G9 default))
(Interpolation)
(Interpolation)
(Feedrate 1500 mm/min)
(Feedrate 1500 mm/min)
(Interpolation)

```
(Interpolation)
```


### 4.22 Inserting chamfers and roundings (G301/G302) (\#FRC/\#CHR/ \#CHF/\#RND)

| G301 | Insert chamfers | (Both functions are effective once <br> between two <br> G302 |
| :--- | :--- | :--- |
|  | Insert roundings | motion blocks.) |

G301 inserts a straight line at the identical angle of inclination to the adjacent contour elements (chamfers).
G302 inserts an arc with a tangential transition to the two adjacent contour elements (roundings).
These functions are non-modal and generate precisely one insertion segment (arc or straight line). G301/G302 blocks may be only written between blocks with active $G$ functions of the group "G00, G01, G02/G03 except for G05".
The I word programmed in the same NC block defines the rounding size or the chamfer width of the insertion segments in [mm, inch]. The I word remains stored and active. This means that I word need no longer be programmed in the following G301/G302 at the same radius or same chamfer width.

In the initial programming of G301/G302, an I word unequal to zero must be programmed in the NC block, otherwise an error message is output (error which leads to abortion of decoding).
Effectiveness of path feedrate in an inserted chamfer or rounding segment:

- If the previous feedrate is G00 (rapid traverse), the segment is also travelled at maximum possible velocity.
- If the previous feedrate is G01/G02/G03, the programmed feedrate is also valid in the segment.
- A feedrate may also be specified in a block containing G301/G302. This feedrate is also valid in all following G01/G02/G03 blocks.
- The feedrate is also adapted when G11 and G41/G42 are active.


## Specific chamfer or rounding feedrate:

Release Note
As of Build V3.1.3057.04, an active feedrate can only be programmed in the inserted chamfer or rounding segment together with a specified chamfer or rounding.

```
#FRC = <feed_rounding_chamfer>
```

(non-modal)

```
<feed_rounding_chamfer>
```

Feedrate in inserted chamfer or rounding segment using the F word unit (e.g. mm/min)

## Programing Example

## Inserting chamfers and roundings (G301/G302)

Chamfer: $90^{\circ}$ corner with 2 straight lines ( $\mathrm{I}=20$ is specified for a chamfer $20 \times 45^{\circ}$ )

| N100 | G00 | G91 | X100 Y0 |  |
| :--- | :--- | :--- | :---: | :---: |
| N110 | G01 |  | Y100 | F200 |
| N120 | G301 | I20 |  |  |
| N130 |  |  | X-60 |  |
| N140 | G00 | G90 | X0 | Y0 |



Rounding: rectangular pocket with corner radius 20 mm , length 200 mm , width 100 mm , segmentspecific feedrates

| Default: |  |  |  |
| :--- | :--- | :--- | :--- |
| N100 | G00 | X0 | Y0 |
| N110 | G01 | X200 | F200 |
| N120 | G302 | I20 | F150 |
| N130 |  | Y100 | F200 |
| N140 | G302 |  |  |
| N150 |  | X0 |  |
| N160 | G302 |  | F200 |
| N170 |  |  | Y0 |
| N180 | G302 |  | F200 |
| N190 |  | X40 | F150 |


| Alternatively: |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: |
| N100 | G00 | X0 | Y0 |  |  |  |  |
| N110 | G01 | X200 | F200 |  |  |  |  |
| N120 | G302 | I20 | \#FRC=150 |  |  |  |  |
| N130 |  | Y100 |  |  |  |  |  |
| N140 | G302 |  | \#FRC=150 |  |  |  |  |
| N150 |  | X0 |  |  |  |  |  |
| N160 | G302 |  | \#FRC=150 |  |  |  |  |
| N170 |  | Y0 |  |  |  |  |  |
| N180 | G302 |  | \#FRC=150 |  |  |  |  |
| N190 |  | X40 |  |  |  |  |  |



The chamfer or rounding is always added in the plane in which the second motion block is programmed.

## Example A:

```
N100 G18 X20
N110 G19
N120 G301 I5
N130 Y20 Z20
```


## Example B:

| N100 | G18 | X20 |
| :--- | :--- | :--- |
| N110 | G301 | I5 |
| N120 | G19 |  |
| N130 | Y20 | Z20 |

Example $A$ and $B$ both yield the same result, contouring in the $Y-Z$ plane.

## Release Note

As of Build V3.1.3057.04 there are additional options to program chamfers and roundings:

## Extended G functions G301 and G302:

Chamfers and roundings are programmed as additional values with G301 and G302. Chamfers and rounding values must always be specified in [mm, inch]; they are non-modal.
G301 and G302 can be programmed directly in the first motion block. A separate NC block is not required.

| G301 $=$ <Fase> | Insert chamfers | (non-modal) |
| :--- | :--- | :--- |
| G302 $=$ <Radius> | Insert chamfers | (non-modal) |

## Programing Example

## Inserting chamfers and roundings (G301/G302)

Rectangular pocket with 2 chamfers (G201) and 2 roundings (G302) with specific feedrates

```
N05 G17 G00 G90 X0 Y0
N10 G01 F2000 X100 G301=20 #FRC=500
N20 Y100 G302=20 #FRC=1000
N30 X0 G301=20 #FRC=500
N40 Y0 G302=20 #FRC=1000
```

N50 X40
N50 M30


## Programming chamfers and roundings with \# commands:

Chamfers and roundings are programmed as additional values with\# commands. Chamfers and rounding values must always be specified in [mm, inch]; they are non-modal.
\# commands can be programmed directly in the first motion block. A separate NC block is not required.
A chamfer can be programmed in two ways: either by specifying

- chamfer width (analogous to G301) or
- chamfer length

| $\# \mathrm{CHR}=<$ Fasenbreite> | (non-modal) |
| :--- | ---: |
| $\# \mathrm{CHF}=<$ Fasenlänge> | (non-modal) |



Rounding is programmed by:
\#RND= <Radius>
(non-modal)


## Programing Example


Rectangular pocket with 2 chamfers (\#CHR, \#CHF) and 2 roundings (\# RND) with specific feedrates

| N05 | G17 | G00 G90 X0 | Y0 |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :--- | :--- |
| N10 | G01 | F2000 | X200 |  | \#CHR=20 | \#FRC=500 |
| N20 |  |  |  | Y100 | \#RND=20 | \#FRC=1000 |
| N30 |  |  | X0 |  | \#CHF=35 | \#FRC=500 |
| N40 |  |  |  | Y0 | \#RND=20 | \#FRC=1000 |
| N50 |  | X40 |  |  |  |  |
| N60 | M30 |  |  |  |  |  |



### 4.22.1 Insert chamfers using G301 as example

With G301 the I word defines the distance between the corner point of the programmed contour and each intersecting point between the inserted straight lines and the contour elements. If one or both the contour elements are arcs, this distance is considered as chord length.

If a given size of the insertion radius or the chamfer results in a direction reversal of one or both contour elements, an error message is output.
:
N10 G91 G01 X80 Y-40 F100 ;P1
N20 G301 I40
N30 G01 X80 Y40 ;P2
:


Fig. 53: Insert a chamfer between two straight lines

```
:
N10 G91 G03 I50 X95 Y-15 F100 ;P1
N20 G301 I30
N30 G03 X80 Y-5 I40 J15 ;P2
```

:


Fig. 54: Insert a chamfer between two arcs

```
\begin{tabular}{ll} 
N10 G01 X20 Y-10 F100 & P1 \\
N20 X20 & ;P2 \\
N30 G301 I30 & \\
N40 X60 Y50 & P3
\end{tabular}
```



Fig. 55: Error due to direction reversal

### 4.22.2 Inserting roundings using G302 as example

With G302 the I word defines the radius of the inserted arc. Its centre point is the intersecting point of the two equidistances with an interval I to the programmed path. The positions of the equidistances are selected so that the programmed contour is retained as far as possible during the contouring operation.

## Notice

No contouring radius can be inserted in tangential block transitions.

```
:
N10 G91 G01 X60 F100 ;P1
N20 G302 I30
N30 X-40 Y-55 ;P2
:
```




Fig. 56: Inserting an arc between two straight lines

```
:
N10 G91 G02 X80 I40 F100 ;P1
N20 G302 I40
N30 G02 X50 I25 J-15 ;P2
```



Fig. 57: Inserting an arc between two circles (angle $\alpha \geq 180^{\circ}$ )


Fig. 58: Inserting an arc between two circles (angle $\alpha<180^{\circ}$ )

### 4.23 Manual mode

Manual mode (HB) permits the external control of single axes with physical elements of manual mode (handwheel, inching keys, joystick). While an interpolation is running, i.e. during the execution of the NC program in HB mode, the operator can add additional set values to the path. The following manual modes are available.

Handwheel function: Any desired path at any desired velocity.
Continuous jog mode:
Incremental
jog mode:
Interruptible
Any desired path at parameterisable velocity.
Pre-specified path at parameterisable velocity
same as incremental jog mode but with the option of motion interruption. jog mode:

These manual modes can be activated from the user interface. The corresponding parameters, e.g. resolution stages, velocity, jogging distance etc., are programmed by using appropriate NC commands ("\#" commands, "G" commands ).
The current manual mode and the axis controlled by a physical manual mode element can be changed at any time. A physical manual mode element can add set values to several axes in several NC channels at the same time. An axis can only be operated in one manual mode and with one operating element.


Fig. 59: Manual mode and its options

### 4.23.1 Selecting/deselecting manual mode with parallel interpolation (G201/G202).

The internal interface between interpolator and manual mode is activated for the programmed path axes. Spindles cannot be programmed.

## Release Note

As of Build V2.11.2010.02 the specification of axes X.. Y.. Replaces the command \#ACHSE [...]. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.

G201 selects manual mode for specific axes. After that, manual mode with parallel interpolation is active for these axes until it is deselected by G202.

G201 <axis_name><dummy_expr> \{<axis_name><dummy_expr>\}

Select manual mode for some specific axes. The coordinate value is only required for syntax reasons; otherwise it is irrelevant. When selected, axes must always be specified.

G202 deselects manual mode for all or specific axes.
G202 Deselect manual mode for all axes (modal, initial state)
... or for specific axes

G202 <axis_name><dummy_expr> \{<axis_name><dummy_expr>\}
<axis_name><dummy_expr>
Deselect manual mode for specific axes. The coordinate value is only required for syntax reasons; otherwise it is irrelevant.

After G202, manual mode offsets accumulated in the interpolator while G201 was active are not deleted. This either occurs

- during the next implicit position adjustment in the channel (e.g. initiated by an axis exchange, transformer selection, etc.) or
- by an explicit set point request with \#CHANNEL INIT[CMDPOS].


## Programing Example

Selecting/deselecting with parallel interpolation(G201/G202)

```
.
G00 X100 Y100
;Status transition of X/Y axes to manual mode)
G201 X1 Y1
P1 = 0
$WHILE P1 == 0
; Set up X/Y axes in manual mode
; ;Program continues by setting P1 to 1 on operating console
$ENDWHILE
; Status transition of all axes to normal mode
G202
;Optional: Request command positions, delete manual mode offsets
#CHANNEL INIT[CMDPOS]
G01 Y200 F500
```


### 4.23.2 Selecting manual mode without parallel interpolation (G200)

The internal interface between interpolator and manual mode is activated for the programmed path axes. Spindles cannot be programmed.


## Release Note

As of Build V2.11.2010.02 the specification of axes X... Y.. Replaces the command \#ACHSE [...]. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.

Manual mode without parallel interpolation is selected with G200.

| G200 | Manual mode for all axes | (non-modal) |
| :--- | :---: | :---: |
| G200 <axis_name><dummy_expr> $\{$ or for specific axes |  |  |

<axis_name><dummy_expr>
Select manual mode for some specific axes. The coordinate value is only required for syntax reasons; otherwise it is irrelevant.

When G200 is programmed, processing of the current NC program is interrupted in the interpolator. Manual modes can be activated, switched over and deactivated. After axes are traversed in manual mode, a request to continue the NC program can be sent to the interpolator by the control statement "Continue motion".
In this operating mode, the offset limits (maximum motion path in manual mode) are automatically set to the software limit switch positions so that the entire range between software limit switches can be traversed in manual mode. After manual mode is deselected, the previous relative offset limits are again valid.

A parallel interpolation of axes during manual mode is not possible with G200. After the "Continue motion" statement, all axes and operating modes are deactivated so that manual mode is no longer possible.

After G200 is deactivated, the decoder requests all manual mode offsets and command positions from the interpolator and sends the current command positions to all participants in the NC channel. Manual mode offsets are stored in the decoder in the variables V.A.MANUAL_OFFSETS (see also Section Axis-specific variables (V.A.) [ 578] and can be addressed in this way in the NC program. Manual mode offsets in the interpolator are deleted.
If G200 is programmed without specifying an axis, this command acts on all existing path axes.

## Programing Example

Selecting without parallel interpolation (G200)

```
G00 X100 Y100
G200 X1 (Manual mode for X axis)
G01 X200 Y200 F600
G01 Y200 F500
G200 (Manual mode for all axes)
G01 Y500 Z500
```


### 4.23.3 Reaction at program end (M02, M30)

At the end of the NC program, the axes may not be traversed in manual mode. This means that the command "M30"or "M02" are treated as if an additional G202 was programmed.

### 4.23.4 Parameterising operating modes

\# commands for parameterisation may only be programmed when manual mode (G202) is deselected for the assigned axes.

### 4.23.4.1 Handwheel mode (\#HANDWHEEL)

## Release Note

As of Build V2.11.2010.02 the command \#HANDWHEEL [...] replaces the command \#SET HR [...]. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.
\#HANDWHEEL [ AX<axis_name> | AXNR<expr> RES1<expr> [ RES2<expr> RES3<expr>] ] (modal)

| AX<axis_name> | Name of manual mode axis |
| :--- | :--- |
| AXNR<expr> | Logical number of manual mode axis, positive integer. |
| RES1<expr>, | Resolution stages (maximum 3), |
| RES2<expr>, | [mm/handwheel rev., inch/handwheel rev., \%/handwheel rev.] |
| RES3<expr> |  |

## Programing Example

Handwheel operating mode

```
When the axis name is specified:
G202 X1
#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
G201 X1
..or when the logical axis number is specified:
G202 X1
#HANDWHEEL [AXNR=1 RES1=0.1 RES2=0.2 RES3=0.5]
G201 X1
```


### 4.23.4.2 Continuous jog mode (\#JOG CONT)

## Release Note

As of Build V2.11.2010.02, the command \#JOG CONT [...] replaces the command \#SET TIP [...]. For compatibility reasons, this command is still available but it is recommended not to use it in new NC programs.
\#JOG CONT [ AX<axis_name> | AXNR<expr> FEED1<expr> [FEED2<expr> FEED3<expr>] ] (modal)

| AX<axis_name> | Name of manual mode axis |
| :--- | :--- |
| AXNR<expr> | Logical number of manual mode axis, positive integer. |
| FEED1<expr>, |  |
| FEED2<expr>, | Velocity stages (maximum 3 ) in $\left[\mathrm{mm} / \mathrm{min}, \mathrm{m} / \mathrm{min}, \mathrm{inch} / \mathrm{min},{ }^{\circ} / \mathrm{min}\right]$ |
| FEED3<expr> |  |

FEED3<expr>

## Programing Example

## Continuous jog mode

```
When the axis name is specified:
G202 X1
#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
G201 X1
..or when the logical axis number is specified:
G202 X1
#JOG CONT [AXNR=1 FEED1=1.0 FEED2=1.5 FEED3=2.0]
G201 X1
```


### 4.23.4.3 Incremental jog or interruptible jog mode (\# JOG INCR)

## Release Note

As of Build V2.11.2010.02, the command \#JOG INCR [...] replaces the command \#SET JOG [...]. For compatibility reasons, this command is still available but it is recommended not to use it in new NC programs.

## \#JOG INCR [ AX<axis_name> | AXNR<expr> DIST1<expr> FEED1<expr>

[ DIST2<expr> FEED2<expr> DIST3<expr> FEED3<expr> ] ]
(modal)

| AX<axis_name> | Name of manual mode axis <br> AXNR<expr> |
| :--- | :--- |
| LIST1<expr> |  |
| FEED1<expr>, | Pairs of (incremental widths; velocity)(maximum 3) in |
| DIST2<expr> | [mm;mm/min, mm;m/min, inch;inch/min,,$\left.^{\circ} ; / \mathrm{min}\right]$ |
| FEED2<expr>, |  |
| DIST3<expr> |  |
| FEED3<expr>, |  |

```
Programing Example
|/>
N10G01 X10
Incremental jog or interruptible jog mode
When the axis name is specified:
G202 X1
#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
G201 X1
..or when the logical axis number is specified:
G202 X1
#JOG INCR [AXNR=1 DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
G201 X1
```


### 4.23.5 Specify offset limits (\#MANUAL LIMITS)



## Release Note

As of Build V2.11.2010.02 replaces the command \#MANUAL LIMITS [...] the command \#SET OFFSET [...]. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.
\#MANUAL LIMITS [ AX<Axisname> | AXNR<expr> NEGATIVE<expr> POSITIVE<expr> ]

AX<Axisname>
AXNR<expr>

NEGATIVE<expr>
POSITIVE<expr>

Name of axis for which the offset limits are valid.
Logical number of axis for which the offset limit is to be valid,
positive integer
Negative relative offset value. Must be programmed as $<0$ in [mm, inch]
Positive relative offset value. Must be programmed as $<0$ in [mm, inch]

This command defines the positive and negative limits for the permissible relative path motion in G201/G202 manual mode for each path axis. The relative negative and positive offset limits refer here to the starting point when manual mode was selected. Offset limits are also considered in G200 by setting the parameter P-CHAN-00114.

## Notice

Relative offset limits can be overwritten at any time in the NC program. A sign check is made. Relative offset limits apply to each axis in the programming coordinate system (PCS).

## Programing Example

## Preset offset limits



Axes that are operated as rotary modulo axis or as spindles (M3, M4, M5, M19, S...) are not considered in this command. These offset limits apply to all manual modes. They can also be overwritten when manual mode is active (G201). Only a sign check of the programmed values is executed. No other checks are made, e.g. checking for range extensions. Relative offset limits can also be specified using the parameter data record P-AXIS-00138 and P-AXIS-00137 of the axis.

### 4.23.6 Example of parameterising an axis in manual mode

## Programing Example

Example of parameterising an axis in manual mode
\%main
\#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
\#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
\#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
\#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]

G201 X1

G202 X1

The command
\#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
specifies the handwheel resolution stages of the $X$ axis. The resolution stages are $0.1,0.2$ and 0.5 $\mathrm{mm} / \mathrm{handwheel}$ revolution.

The command
\#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
specifies the motion velocities for continuous jog mode of the $X$ axis. The velocity stages are 1.0 , 1.5 and $2.0 \mathrm{~mm} / \mathrm{min}$.

The command
\#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]
parameterises incremental jog and interruptible jog mode for the $X$ axis. Incremental widths and velocities are programmed in pairs.

Stage 1: Incremental width $0.1 \quad$ velocity $1.0 \mathrm{~mm} / \mathrm{min}$ mm ,
Stage 2: $\quad$ Incremental width $0.2 \quad$ velocity $1.5 \mathrm{~mm} / \mathrm{min}$ mm ,
Stage 3: Incremental width $0.5 \quad$ velocity $2.0 \mathrm{~mm} / \mathrm{min}$ mm ,

The command
\#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]
Sets negative offset limit for manual mode to -5 mm and the positive offset limit to +5 mm .
The following commands select or deselect manual mode for the $X$ axis. If G 202 is programmed along, manual mode is deactivated for all axes.

## G201 X1

G202 X1
G202

### 4.24 Requesting offset, command and actual values

The NC commands described below trigger jobs to include manual mode offset, command and actual values of the interpolator in the operating data of the NC program interpreter. This data can then be accessed by variables in the NC program.
Offset, command and actual values of path axes can only be requested. Axes that are operated as rotary modulo axis or spindles (M3, M4, M5, M19, S...) are not considered in the following commands.
Synchronisation between the NC program interpreter and the interpolator is identical for all the NC commands described. The job is only complete after all the data is adopted in the working data of the NC program interpreter and decoding of the NC program is continued.

Overview of commands and properties:

| Command | Channel | Manual | Updating of ... |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| initialisa- <br> tion <br> mode off- <br> sets in IPO | V.A.MANUA <br> L_OFFSETS | V.A.ABS | P parameter, <br> V.S./ V.P./ <br> V.L. |  |  |
| \#GET MANUAL OFF- <br> SETS | no | are retained | yes | no | no |
| \#CHANNEL INIT [CM- <br> DPOS] | yes | are deleted | no | yes | no |
| \#CHANNEL INIT <br> [ACTPOS] | yes | are deleted | no | yes | no |
| \#GET CMDPOS | no | are retained | no | no | yes |
| \#GET ACTPOS | no | are retained | no | no | yes |

### 4.24.1 Request current manual mode offsets and file to "V.A.MANUAL_OFFSETS[ ]" (\#GET MANUAL OFFSETS)

## Release Note

As of Build V2.11.2010.02 replaces the command \#GET MANUAL OFFSETS the command \#GET IPO OFFSET. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

## \#GET MANUAL OFFSETS

This command is useful when combined with G201/G202 (manual mode with parallel interpolation). The NC program interpreter requests the current manual mode offsets of all path axes assigned to it by the interpolator. After the values are supplied, they are stored in the NC program interpreter in the variables V.A.MANUAL_OFFSETS (see also Section Axis-specific variables (V.A.) [ $>578]$ ) and can then be addressed in the NC program. This is no implicit position initialisation of the NC channel. Manual mode offsets are not deleted in the interpolator. The variables V.A.ABS (see also Section Axis-specific variables (V.A.) [> 578]) are not updated.

## Programing Example

\#GET MANUAL OFFSETS

```
X100
G201
..... Moving in manual mode
G202
#GET MANUAL OFFSETS
G01 X[100 + V.A.MANUAL_OFFSETS.X] F500 (Position initialisation of the)
(NC channel after)
(deselecting manual mode)
```


### 4.24.2 Request current command positions and file to "V.A.ABS[ ]" (\#CHANNEL INIT)

## Release Note

As of Build V2.10.1504 the command \#CHANNEL INIT [CMDPOS] replaces the command \#SET DEC LR SOLL. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

## \#CHANNEL INIT [CMDPOS \{ AX<axis_name> | AXNR<expr> \} ]

| AX<axis_name> | Name of the axis for which the command value is requested. |
| :--- | :--- |
| AXNR<expr> | Logical number of the axis for which the command value is requested, |
|  | Positive integer. |

The NC program interpreter requests the interpolator for the current command positions of all or specific path axes, files them to the working data and initialises the NC channel with these positions. The current command positions in the working data can then be accessed by programming the variable V.A.ABS (see also Section Axis-specific variables (V.A.) [ 578]). Any stored manual mode offsets in the interpolator are then deleted automatically. The axis positions correspond to those for zero offset. The values stored in the variable V.A.MANUAL_OFFSETS (see also Section Axis-specific variables (V.A.) [ 578]) are not updated.

## Release Note

*The axis-specific request for command positions is available as of Build V2.11.2038.03.

Notice
If no axes are programmed, the command positions for all path axes in the channel are requested. If axes are programmed, only the command positions for them are requested.
$\qquad$


```
# CHANNEL INIT [CMDPOS AXNR=1 AXNR=2 ] (..or logical axis number..)
# CHANNEL INIT [CMDPOS] (..or for all path axes)
#MSG ["Cmdpos X:%F, Y:%F ",V.A.ABS.X, V.A.ABS.Y]
```

M30

### 4.24.3 <br> Request current actual positions and file to "V.A.ABS[ ]" (\#CHANNEL INIT)

## \#CHANNEL INIT [ ACTPOS \{ AX<axis_name> | AXNR<expr> \} ]

AX<axis_name>
AXNR<expr>

Name of the axis for which the actual value is requested.
Logical number of the axis for which the actual value is requested,
Positive integer.
The NC program interpreter requests the interpolator for the current actual positions of all or specific path axes, files them to the working data and initialises the NC channel with these positions. Access can then be made to the current actual positions in the working data by programming the variable V.A.ABS (see also Section Axis-specific variables (V.A.) [ 578]). Any stored manual mode offsets in the interpolator are then deleted automatically. The axis positions correspond to those for zero offset. The values stored in the variable V.A.MANUAL_OFFSETS (see also Section Axis-specific variables (V.A.) [> 578]) are not updated.

## Notice

If no axes are programmed, the actual positions for all path axes in the channel are requested.
If axes are programmed, only the actual positions for them are requested. Command positions are requested for axes which are not programmed.


## Attention

The axis can be moved by adopting the actual position as command position. If an axis is adopted in an NC program loop, it may cause the drive to drift.


### 4.24.4 Request current command positions of axes and file to variables or parameters (\#GET CMDPOS)

## Release Note

As of Build V2.11.2010.02 replaces the command \#GET CMDPOS [...] the command \#SET IPO SOLLPOS [...]. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.
\#GET CMDPOS [ V.S.<string> | V.P <string> | V.L<string> | P<expr> = <axis name>
\{ ,V.S. <string> | V.P <string> | V.L<string> | $\mathbf{P}$ <expr> = <axis_name> \} ]

The NC program interpreter requests the interpolator for the current command positions of the specified path axes and files them to the specified self-defined variables (V.S./ V.P./ V.L.) or parameters (P..). This is no implicit position initialisation of the NC channel. Any manual mode offsets of the selected axes in the interpolator are not deleted. The variables V.A.ABS and V.A.MANUAL_OFFSETS are not updated.

## Programing Example

\#GET CMDPOS

```
.....
#GET CMDPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1 CV.P.POS2
```


### 4.24.5 Request current actual positions of axes and file to variables or parameters (\#GET ACTPOS)

## Release Note

This command is available as of CNC Build V2.11.2022.05 onwards.
\#GET ACTPOS [ V.S.<string> | V.P <string> | V.L<string> | $\mathbf{P}$ <expr> = <axis name>
\{ ,V.S. <string> | V.P <string> | V.L<string> | $\mathbf{P}<$ expr> = <axis_name> \}]

The NC program interpreter requests the interpolator for the current actual positions of the specified path axes and files them to the specified self-defined variables (V.S./ V.P./ V.L.) or parameters (P..). This is no implicit position initialisation of the NC channel. Any manual mode offsets of the selected axes in the interpolator are not deleted. The variables V.A.ABS and V.A.MANUAL_OFFSETS are not updated.

Programing Example
\#GET ACTPOS
$\qquad$
\#GET ACTPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1 CV.P.POS2

### 4.25 Gear change (G112)

| G112 | Gear change | (non-modal) |
| :--- | :--- | :--- |

The function G112 can change gear stages for individual path axes. G112 is also available for spindle axes. However, the special characteristics described in Section Gear change (G112) [ $\downarrow$ 666] must be taken into consideration.

With axes which have no real gear, this functionality can be used to change various axis parameters. For example, controller gain and position lag monitoring can be matched to one another in a separate parameter block and can then be selected by the "gear number" defined in the NC program.

## Attention

G112 may not be programmed with a path condition in the same NC block.

## Programing Example

Gear change (G112)

N50 G112 X4 Y8 (X axis: gear stage 4, Y axis: gear stage 8)

### 4.26 <br> Influence on the look-ahead functionality (G115/G116/G117)

G115 General influence on the look-ahead functionality
G116 Influence on the calculation of block transition velocity
G117 Reset the look-ahead functionality
(modal)
(modal)
(modal, initial state)

## Overview of the functionality of the look-ahead function

Various single functions are implemented in the look-ahead function. They limit motion velocities to maximum values in order to always maintain the permissible axis velocities and axis accelerations.

The following single functions influence the velocity profile:

- (1) Calculation of maximum path velocities based on the maximum axis velocities of the axes participating in the motion as specified in the channel parameter list P-CHAN-00071. The velocity on the programmed path is limited so that each of the axes participating in the motion does not exceed its maximum axis velocity.
- (2) Calculation of maximum path velocities on curved paths based on maximum chord error. In this case, a curve radius can be determined on each curve at the point of greatest curvature. A circular section can then be defined on this circle and its starting and end points correspond to a point generated on the path. Provided that the real mechanical system is moved along the associated chord, the height of the associated circular section can be included as an error criterion.
- (3) Calculation of maximum path velocities on curved paths based on the maximum axis acceleration specified in the channel parameter list P-CHAN-00071. The maximum path velocity calculated based on the secant error can still be too high, particularly with a short tracing time and smaller circular movements. The resultant centripetal accelerations would then exceed the permissible axis accelerations.
- (4) Calculation of path transition velocity based on the block transition geometry, taking into account the axes participating in the motion and the limit values specified in the channel parameter list P-CHAN-00071. As a rule, the relevant portions of the single axes alternate in the total motion with a block transition. However, this change in axis portions only occurs in the first tracing cycle on the new path (for linear motion). If it is assumed that the transition should take place without the generated acceleration (e.g. by the slope function), the resultant axis accelerations can be estimated. The maximum path velocity at block transition can then be calculated so that the maximum axis accelerations are not exceeded at the transition.


## General case:

## Normal mode with look-ahead

After program start all functions of the look-ahead function are switched on (G117). This ensures that the parameters specified in the channel parameter list P-CHAN-00071, e.g. maximum axis velocity and maximum axis acceleration, are always maintained.

## Special case:

Switch off single functions of the look-ahead function by G115

## Programing Example

Influence the look-ahead functionality (G115/G116/G117)

Nn G115 = $0 \quad$ (Switch off the single functions (2), (3) and (4))
(of the look-ahead function)

The table below explains the possible combinations of the single functions.

## Notice

The corresponding selected or deselected functions always refer to all axes.

Overview of permissible identification numbers (ID) related to G115:


## Programing Example

Nnn G115 = 2 (Switch off single functions (2) and (3))
Nnn G115 = 12 (Switch off single functions and (4))
Nnn G115 $=14$ (Switch off single functions after)
(program start, cf. G117)

## Special case:

Influence the look-ahead functionality with G116, G117
The G116 function disables the calculation of the block transition velocity (single function (4)) for single programmable axes. No reduction in block transition velocity (path velocity) takes place because of "corners" in the path.

## Notice

In this case, the dynamic axis data (axis accelerations) are not normally maintained.

## Programing Example

```
(No reduction in block transition)
    (rate of different axis)
    (rates of the axes X or Y)
N20 G01 G91 X100 Y-100 F1000
N30 X-100
```

In this example, block transition N20/N30 is traversed without any drop in set velocity. The command values change at the block transition in jumps according to the axis portions on the path.

## Attention

The coordinate value is only required for syntax reasons; otherwise it is irrelevant.

The G117 function re-enables all single functions of the look-ahead-function (default setting after program start).

## Programing Example

## $4.27 \quad$ Override (G166)

G166 Set override to $100 \%$ (non-modal)

The G166 function switches off the external influence on path axis overrides blockwise and suppresses the effect of programmed override values on the path [ $>460$ ] or an axis [ 790].

## Programing Example

## Override (G166)

```
%override_G166_extern
;Assumption: External override 50%
N10 G00 G90 X0 Y0 Z0 ;Rapid traverse 50%
N20 G01 X10 F2000 ;Feed F1000
N30 X20 Y20 ;Feed F1000
N40 G166 Z30 ;Feed F2000, override 100%
N50 X30 ;Feed F1000
N60 G166 Y30 ;Feed F2000, override 100%
N70 G166 G00 X0 Y0 Z0 ;Rapid traverse 100%
N80 G01 F3000
N90 X10 Y20 Z30 ;Vorschub F1500
M30
MN10 G00
o%override_G166_path
;Path override: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]
override_G166_path
; Path override: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]
N20 G01 X10 Y10 Z10 F1000 ;Feed F1200
N30 G166 X20 Y20 Z20 ;Feed F1000, override 100%
N50 G00 X50 ;Rapid traverse 75%
N60 G166 Y50 Z50 ;Rapid traverse 100%
M30
%override_G166_ax
;Axis override for X: G01 20%, G00 60%
N10 X[OVERRIDE FEED_FACT=20 RAPID_FACT=60]
N20 G00 X10 ;Rapid traverse in X at 60% override
N30 Y10 Z10 ;Rapid traverse in Y/Z at 100%
N40 G166 X20 ;Rapid traverse in X at 100% override
N50 G01 X30 F2000 ;Feed in X with F400, 20% override
N60 Y20 Z20 ;Feed in Y/Z with F2000 (100%)
N70 G166 X40 ;Feed in X with F2000, 100% override
M
```


### 4.28

Cycle synchronisation at block end (G66)
G66

The G66 function adapts the velocity in the motion block so that each interpolation point is always reached in the interpolator (IPO) cycle.
A motion block programmed in combination with G66 can be executed reproducibly in program loops without interference or undesirable velocity fluctuations.

## $4.29 \quad$ Rotate the coordinate system in the plane (G68/G69)

## Release Note

This function is available as of CNC Build V3.1.3079.33.

This function rotates a coordinate system in the current plane (G17/G18/G19). Contours programmed in the machine coordinate system can be adapted quickly and easily to workpieces in offset positions.

Contour rotation acts directly on the programmed axis coordinates (contour) before all other con-tour-influencing functions, i.e. all offsets and mirroring operations are not influenced by the rotation and can be used as before (*).
Rotation may also be applied within an already rotated coordinate system (\#(A)CS).
A change of plane with $\mathrm{G} 17 / \mathrm{G} 18 / \mathrm{G} 19$ automatically deselects an active contour rotation and a warning is output.

As a alternative to G68/ G69, contour rotation can be programmed using \#ROTATION [> 398].

Syntax (example in G17):

| G68 R.. X.. Y.. | Select contour rotation <br> Deselect contour rotation |
| :--- | :--- |
| R.. | Rotation angle in degrees $\left[{ }^{\circ}\right]$, absolute. If no angle is specified, the value $0^{\circ}$ is set. The <br> rotation angle has no influence on already programmed circle radii. |
| X... Y.. | Absolute coordinates of the centre of rotation in $[m m, ~ i n c h] ~ i n ~ t h e ~ m a i n ~ a x e s ~ o f ~ t h e ~ c u r-~$ <br> rent plane. |
| The following applies: G17 -X and $\mathrm{Y}, \mathrm{G} 18-\mathrm{Z}$ and $\mathrm{X}, \mathrm{G19}-\mathrm{Y}$ and Z <br> With coordinates that are not programmed, the current actual position is set as the <br> centre of rotation. |  |



Fig. 60: Significance of rotation parameters in the main plane (example G17):

| $a: X .$. | $b: Y .$. | $c: R .$. |
| :--- | :--- | :--- |

The programmed rotation parameters can be read with the following variables:
V.G.ROT_ACTIVE Supplies the value 1 if a rotation is active
V.G.ROT_ANGLE Rotation angle
V.G.ROT_CENTER1 Offset of the first main axis relative to the centre of rotation
V.G.ROT_CENTER2 Offset of the second main axis relative to the centre of rotation

## Notice

(*) It makes no difference whether the offsets (e.g. G54, G92 etc. ) were programmed before or after G68; they always act in the axis directions of the basic coordinate system of the machine (MCS).
In addition, tool offsets always act independently of P-TOOL-00010 in the axis directions of the MCS.

## Programing Example

Rotation in a plane (contour rotation)

```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29
```

\%ang1.nc
N100 G53 G17
N110 LL part
N130 G68 R-45 X10 Y100
N140 LL part
N150 G21 (mirror X coordinates)
N160 LL part
N170 G69
M30


## $5 \quad$ Switching and supplementary functions (M/H/T)

A complete list of G functions is contained in the overview of commands in the Appendix under M functions (M..) [〉 806].

### 5.1 User-specific M/H functions

M/H functions are mostly determined by programming the "Programmable Logic Controller" (PLC). This controller contains an interface ("Window to PLC") information on the programmed M/ $H$ functions in the current NC block. In addition, it provides synchronisation conditions which control synchronisation between the NC and the PLC.

```
M<expr>
or
H<expr>
```

M/H functions can be programmed with general mathematical expressions. In particular, this allows the assignment of values by means of parameters, e.g. MP10. Before access to the M/H function, the calculated numerical value is rounded off and converted into an integer. The smallest permissible value is 0 (zero). Negative values generate an error message.

The maximum number of M/H functions per NC channel [6] [ [ 819]-8.1/-8.2 and the maximum number of M/H functions per NC block [6] [ 819]-8.3 are fixed.
The EXIST function (see Section Arithmetic expressions <expr> [> 31] checks whether a variable exists at all.

Programing Example
User-specific M/H functions

The EXIST request first checks whether a user-specific $M$ function is at all defined in the channel parameter list $\mathrm{P}-\mathrm{CHAN}-00027$ before the M function is executed.

```
N10 G90 Y0
N20 $IF EXIST[M80] == TRUE
N30 X0 Y0 Z0 M80 (M80 is available)
N40 $ELSE
: ..
N60 $ENDIF
```

M30

The following M functions have a predefined meaning in the NC program, i.e. they are not freely available. The user can only assign their synchronisation mode P-CHAN-00027.

| M00 | Programmed stop |
| :--- | :--- |
| M01 | Optional stop |
| M02 | Program end, stop the machine |
| M10, M11 | Latch axes |
| M17 | Subroutine end |
| M29 | Subroutine end |
| M30 | Program end, stop the machine |

### 5.1.1 Programmed stop (M00)

M00 causes an interrupt in a running NC program, e.g. to execute a measurement or to remove chips. Machining continues when the "CONTINUE MOTION" button is pressed. If MOO is present in a block with motion statements, the interruption occurs after the motion statement is executed.

### 5.1.2 Optional stop (M01)

M01 acts in the same way as M00. However, the effectiveness of M01 must be activated on the operating console of the controller.

### 5.1.3 Program end (M02/M30)

M02 and M30 are identical. In the main program and subroutine, they cause a reset to the start of the main program and reset the NC to initial state.

In addition, a signal can be sent to the PLC by configuration. For example, the PLC uses this signal to synchronize a workpiece change.

### 5.1.4 Subroutine end (M17/M29)

M17 and M29 terminate a subroutine properly. See also Section Subroutine techniques [> 204]
A signal can also be forwarded to the PLC by configuration.

### 5.1.5 Call a tool change program (M06)

When configured as required (P-CHAN-00118), the M function M06 is not interpreted as a technology function. Instead, it calls a global subroutine. In this subroutine the user can program all the actions required for a physical tool change.

## Notice

Depending on the version, the processing lines of the tool change program are masked or visible in the display when M6 is executed in the default setting while the NC program is running or in single block mode cycle. When active display is off, only the M6 call is displayed during this time. This feature is switchable by the channel parameter P-CHAN-00211.

Notice
In TwinCat systems, all cycle lines are visible in the display by default.

### 5.2 Axis-specific M/H functions

In general, user-specific M/H functions programmed in conventional DIN syntax are handled and executed for specific channels.

If the user wants to force axis-specific handling for each user-specific M/H function, he can also preassign them in the channel parameters P-CHAN-00039 and P-CHAN-00025.

```
m_default_outp_ax_name[<m_expr>] <axis_name>
or
h_default_outp_ax_name[<h_expr>] <axis_name>
```

<m_expr> User-specific M function with axis-specific effect.
<h_expr> User-specific H function with axis-specific effect.
<axis_name> Axis name of the axis on which the M/H function is to act. Path axes and spindle axes are per-
mitted in this case.

## Programing Example

## Axis-specific M/H functions

Channel parameter list [1] [ 819]:
:
\# Definition of axis-specific M functions
\# ================================================120
m_default_outp_ax_name[20] S2
m_default_outp_ax_name[21] S3

```
m_default_outp_ax_name[22] Z
:
# Definition of axis-specific H functions
# ===============================================
h_default_outp_ax_name[10] X
h_default_outp_ax_name[11] Y
h_default_outp_ax_name[12] Z
:
NnS1000 M3 M20 M21 M22 H10 (S and M3 act on the main spindle axis,
    M20 acts on the S2 spindle axis
M21 acts on the S3 spindle axis and
    M22 acts on the Z axis
    H10 acts on the X axis)
```

:

Notice
If an axis-specific $M / H$ function is programmed in a spindle-specific bracket expression, the default setting is ignored and the $\mathrm{M} / \mathrm{H}$ function acts on the corresponding spindle axis.

Moreover, for path axes, the NC program can also force an axis-specific output of M/H functions by the use of bracketed expressions:

```
<axis name> [ M<expr> H<expr> { M<expr> H<expr> } ] {<axis name> [... ]}
```

<axis_name> Axis name of the axis on which the M/H function is to act. Only path axes are permitted in this case.
$\mathrm{M}<e \operatorname{expr} \quad \quad$ User-specific M function with axis-specific action.
H<expr> User-specific H function with axis-specific action.

## Programing Example

:
N10 S1000 M3 X100 X[M20 H12] Y[H10] (S, M3 act on the main spindle axis)
(M20, H12 act on the X axis)
(H10 acts on the Y axis)
:

This gives the user the option of executing his specific $\mathrm{M} / \mathrm{H}$ functions very flexibly by programming or configuring for specific axes or specific channels.

## Programing Example



## $5.3 \quad \mathrm{M} / \mathrm{H}$ functions with optional additional information

With $\mathrm{M} / \mathrm{H}$ functions, an additive value can be optionally programmed in the NC program and made available to the PLC on the technology interface in combination with the M/H function.

The additive value can be programmed in all user-specific $M / H$ functions and in the internal $M$ functions M00, M01, M02, M17, M29 and M30. The M/H functions can then be used without restriction in both channel-specific and axis-specific programming syntax.

The following $M$ functions may only be programmed without an additive value:

- spindle functions M03, M04, M05 and M19
- gear change functions M40-M45 and
- the subroutine function M6 (if configured as such in P-CHAN-00118)

```
M<expr> [ = <additive_expr> ]
or
H<expr> [ = <additive_expr> ]
```

M<expr>
H<expr>
<additive_expr>

User-specific M function
User-specific H function
Optional additive value. It can be programmed directly as a negative or positive integer or as a general mathematic expression.

## Programing Example

## M/H functions with optional additive information

```
%m_h_add_fct
(M functions with additive value)
N10 M52=-345
N20 M12=123 (with channel parameter m default outp ax_name[12] Z)
N30 M10=321 (with channel parameter m_default_outp_ax_name[10] S)
N35 P1 = 567 P2 = 345
N40 X[M54=P1]
N50 S[REV 1000 M03 M63=-789]
N60 M12=123 M10=321 M52=-345 X[M54=567] S[REV 1000 M03 M63=-789]
N70 M63=-789 M52=-P2 M54=567
N80 X[M52=-345 M54=567] Y[M63=-789] S[M05 M63=789 M54=-567] M54 M63
```

```
(H functions with additive value)
N110 H5=-345
N120 H6=123 (with channel parameter h_default_outp_ax_name[6] Z)
N130 H9=321 (with channel parameter h_default_outp_ax_name[9] S)
N135 P3 = 567 P4 = -345
N140 X[H7=P3]
N150 S[REV 1500 M04 H8=-789]
N160 H6=123 H9=321 H5=-345 X[H7=567] S[REV 1500 M04 H8=-789]
N170 H8=-789 H5=P4 H7=567
N180 X[H8=-789 H4 H5=-345] Y[H7=567] S[M05 H5=345 H7=567] H3 H8
(Mixed M/H functions with additive value)
N200 X[M52=-345 H4 H8=-789 M54=567 H5=345] H3=333 M54=444 H7=567 M63
(M/H functions with additive value in axis-specific function (INDP))
N05 X[INDP G90 G01 FEED=2000 POS=555 M54=151 H8=-181]
N999 M30=111
```

This additive information can be read in the PLC via the data of the $\mathrm{M} / \mathrm{H}$ function.
See also [HLI//data of the M/H function]

### 5.4 Tool position selection (T)

$\mathbf{T}$ <expr> $\quad$ Select too place $\quad$ (modal)

The tool command determines the tool required for the machining step. The tool number is forwarded externally to the PLC which loads the selected tool to a tool magazine for a tool change. It should be noted that the T word itself has no influence on the internal controller calculation of tool geometry (tool data). The D word [ ${ }^{\text {4 492] }}$ is used for this purpose. However, it is activated automatically by the T word if P-CHAN-00014 is programmed accordingly.

The T word initiates no tool change. Normally, this takes place with machine function M06.
Tool data [5] [> 819], [6] [ $>819]-9.7$ is stored ion the tool data table of the decoder [6] [ 819]-9.1. In addition, there is also the alternative option of communicating with a decentralised (external) tool data management system provided by the user and requesting tool data via an additional interface. The internal or external access to tool data is parameterised P-CHAN-00016.
In order to avoid waiting periods between the time that tool data is loaded and then transferred, the tool to be externally loaded normally requests one or more motion blocks before the tool data is actually included in the calculation P-CHAN-00087. If several possibly different T numbers are programmed before the D number of the first corresponding T number appears, all T numbers before the last one programmed are ignored. This means that the data supplied by the external tool management system is deleted if the tool $D$ number does not coincide with the $T$ number programmed.

## $6 \quad$ Velocities (F/E)

| F <expr> | Feedrate within the block | (modal) |
| :--- | :--- | ---: |
| E <expr> | Feedrate at block end | (non-modal) |

With interpolation modes G1 [> 54], G2, G3 [ 55] or for G100 [> 98], the programmed paths are traversed at the path velocity agreed in the $F$ word. The $F$ and $E$ words are specified

- in [mm/min, m/min, inch/min] for translatory axes
- in [ $\% / \mathrm{min}$ ] for rotary axes.

The F word is modal. Programming in $\mathrm{mm} / \mathrm{min}$ or $\mathrm{m} / \mathrm{min}$ can be configured with the channel parameter P-CHAN-00108.

The command G93 [ 157] can also specify a machining time by the F word instead of a path velocity. The description in contained in the Section "G functions".
The path velocity at block end is programmed by the E word. The value is block-specific, i.e. nonmodal. If the $E$ word is not specified or the programmed value is greater than the $F$ word, the $E$ word is assigned the value of the F word

## Notice

The E word is only effective in combination with G94 [> 157].
Due to active contour-changing functions such as G301, G302 [ 158], G41, G42 [ 498], G261 [ $>$ 127], \#HSC [ $>250$ ] [OPMODE 1...], inserted contour elements adopt the E feedrate of the main blocks as the new F feedrate.

For all other spline and HSC functions, it is recommended to avoid the use of the E feedrate.

Values can be assigned to the $F$ and $E$ words either directly or for each parameter. In this case, decimal numbers are also permitted (REAL format).

## Notice

The PLC can also specify path feed externally and also weight it with the NC command \#FF [ ${ }^{\text {483]. }}$

## Programing Example

Fig. 61: Program feedrate using F word

## Programing Example



Fig. 62: Program feedrate using F and E words

## Programing Example

```
N10 G01 G94 G90 G261
N20 X100 F1000 E200 ; (F feedrate }1000\textrm{mm}/\textrm{min},\textrm{E}\mathrm{ feedrate 200mm/min
N30 Y100 E200 ; F feedrate 1000 mm/min, E feedrate 200mm/min
N40 X0 ;F feedrate 1000 mm/min
N50 Y0 E200 ; F feedrate 1000 mm/min, E feedrate 200mm/min
N60 X50
N70 G260
```

Vorschub


Fig. 63: Effect of E word on inserted contour elements (here G261)

## $7 \quad$ NC block numbers (N)

The NC block number can be programmed with the address letter "N". If configured, this number is entered both in the display data as well as in the error messages (alternatively the offset value in the file can also be used for this purpose).

N <expr>

The block number can be programmed using general mathematical expressions. In particular, this permits value assignment by parameters e.g. the run parameter in program loops. The calculated numerical value is automatically rounded off internally and converted into an integer.

## Programing Example

## NC block numbers ( N function)

N10 G01 X200 F500
N20 \$FOR P1=1, 10, 1
N[P1*100] XP1
-
N30 \$ENDFOR

For program flow, the block number has no significance. Therefore, it need not be used in ascending form.

## 8 Subroutine techniques

Identical motion sequences and function flows that are repeated several times can be implemented as subroutines. There are two types of subroutines:

1. Local subroutines
2. Global subroutines or cycles

Several subroutines (local or global) nested in one another can be called. The number of subroutines used and the nesting depth are fixed [6] [ 819]-24.

Variables are transferred from the main program to the subroutine indirectly using P parameters or directly in cycles by calling specific transfer parameters (@P...).

## Overview of program definition and program call:

Main program

| Definition: | \%PROG_NAME or \% PROG_NAME | The main program definition is only required <br> when there are local subroutines. |
| :--- | :--- | :--- |
| Call: | From the operating console by specifying the <br> complete file name (including file extension) |  |

Local subroutine

| Definition: | \%L UP_NAME | Separator between "L" and program defini- <br> tion, otherwise it is assumed that a main pro- <br> gram is defined |
| :--- | :--- | :--- |
| Call in NC pro- <br> gram: | LL UP_NAME | Separator between "LL" and program call, <br> otherwise it is assumed that a global sub- <br> routine is called |

Global subroutine

| Definition: | \%PROG_NAME or \% PROG_NAME | A subroutine definition is only required if there <br> are local subroutines |
| :--- | :--- | :--- |
| Call in NC pro- <br> gram: | L FILE_NAME | Specify the complete file name (including file <br> extension). Separator between "L" and pro- <br> gram call |

## Notice

If the first character that is neither a separator nor a "\%" is found in the file outside the comments, this character is evaluated as the first character of an unnamed main program. It also means that no block numbers may be programmed in front of "\%".

### 8.1 Local subroutines (Call LL <string>) <string>)

A local subroutine is called by

LL <string> (Caution: Blank character between "LL" and <string> ... is mandatory).
<string> Name of local subroutine without quotation marks, maximum 83 characters

Local subroutines (LSUB) together with the main program are located in a common data file which must state all the subroutines before the actual main program.

It should be noted that local subroutines can only be called from the main program in the same data file.
Local subroutines begin with "\%L" and a program designation. Between "L" and the program designation, there must be a minimum of one separator.

The end of the subroutine is marked by the functions M17 or M29. A program abort is also possible with M02 or M30. A warning is then issued.

If M17 or M29 is missing, the subroutine is ended by "\%" (first character of the following main program of a further subroutine).

## Programing Example

Local subroutines (Call LL <string>) <string>)

```
Structure of a data file consisting of the NC main program and local subroutines:
%L UP1
N1 .....
N2 .....
N9 M17
%L UP2 
%I UP2 
%I UP2 
N19 M29 (M29 can also be left out)
%MAIN
N100 .....
N105 .....
N200 LL UP1 (Call of 1st LSUB)
N250 LL UP2
N300 M30
    (1st local subroutine)
    (M17 can also be left out)
    (Main program)
    (2nd local subroutine)
``` rule that the sequence in the NC block has no significance for program execution (see Section NC block structure [> 24]).

\subsection*{8.2 Global sub-routines (Call L <string>)}

A global subroutine is called by

\section*{L <string>}
<string> File name of the global subroutine without quotation marks, max 83 characters (including file extension and possibly an absolute or relative path name)

Global subroutines (GSUB) are independent program units in a separate file. A global subroutine is called by its complete file name (including file extension). A global subroutine can also consist of local subroutines and a main program part.

It is only necessary to specify the name of the global subroutine by \% in the file to indicate the start of the main program part after defining the local subroutines. It can be skipped if the file contains no local subroutines.

The calling main program is also stored as an independent program unit in another file. Global subroutines can be called form all main programs.

The end of a global subroutine is marked by the functions M17 or M29. A program abort is also possible with M02 or M30. A warning is is then issued.

\section*{Programing Example}

Global subroutines (Call L <string>)
```

;all global subroutines
%MAIN (main program)
N100 .....
N105 .....
N200 L gup_1.nc ;Direct call of a global subroutine via
;File name if program path is configured
N250 L D:\prog\ini_1.nc ;Call a global subroutine by
;Specify absolute program path
N300 M30

```

\subsection*{8.3 Parametric subroutine call (LL / L V.E. or macro)}

Instead of using fixed names, local and global subroutines can also be called by external variables or macros. This permits a parametrisable flow of the NC program.

External variables of the string or string array type (see also Section External variables (V.E.) [ 615]). The maximum string length of local subroutines is 83 characters. It is also 83 characters for global subroutines but this includes an absolute or relative path name.

Macros must be defined by L or LL before they are used. The macro content has a maximum string length of 80 characters.

A local subroutine is called by

LL V.E. ... (Caution: Blank character between "LL" and V.E.... is mandatory).
or
LL "<Makroname>" (Caution: Blank character between "LL" and macro name is mandatory).
\begin{tabular}{ll} 
V.E. ... & Name of local subroutine parameterised by external variable \\
"<macro name>" & \begin{tabular}{l} 
Name of local subroutine parameterisable by macro. If the macro is not defined, the <macro \\
name> is treated as a normal local subroutine name.
\end{tabular}
\end{tabular}

A global subroutine is called by
```

LV.E. ... or L V.E. ..
or
L"<Makroname>" or L "<Makroname>"

```
V.E. ... Name of the file in which this global subroutine is stored when the file is parametrised by an external variable.
"<macro name>" Name of the file in which this
Subroutine is stored when the file is parametrised by macro. If the macro is not defined, the <macro name> is treated as a normal global subroutine name.

\section*{Programing Example}

\section*{Parametric subroutine call (LL / L V.E. or macro)}

Call of subroutines by "string"-type external variables
```

;local subroutine
%L TASCHE
N10 .....
N99 M17
;Main program
%MAIN
N100 .....
N105 .....
;Call of local subroutine by ext. Variable V.E.LUP,
;which contains the string TASCHE
N110 LL V.E.LUP

```
; (Call of global subroutine via ext. Variable V.E.GUP,
; which contains the string of a file name
N200 L V.E.GUP

\section*{Programing Example}

Parametric subroutine call (LL / L V.E. or macro)
Call of subroutines by macros
```

;local subroutine 1
%L TASCHE 1
N10 .....
N99 M17
;local subroutine 2
%L TASCHE 2
N10 .....
N99 M17
;local subroutine 3
%L TASCHE 3
N10 .....
N99 M17
;Main program
%MAIN
;Macro definitions
N10 "LUP_1" = "TASCHE_1"
N20 "LUP 2" = "TASCHE 2"
N30 "LUP_3" = "TASCHE_3"

```
```

N40 "GUP_1" = "gup_1.nc"
N50 "GUP_2" = "D:\prog\ini_1.nc"
N100 .....
;Call of local subroutines by macros,
;which contain the strings TASCHE_1, TASCHE_2,
;TASCHE 3
N110 LL "LUP_1"
N120 LL "LUP 2"
N130 LL "LUP_3"
N200 .....
;Call of global subroutines by macros
;which contain file name strings
N210 L "GUP_1"
N220 L "GUP_2"
N300 M30

```

\subsection*{8.4 Implicit global subroutine call at program start}

The same recurring initialisations of data must often be executed at the start of different NC main programs. If these initialisations are summarised in a global subroutine, this program can be executed implicitly as the first action at every NC program start with the channel parameter P-CHAN-00119.

The full scope of NC programming functions can be used in this global subroutine. This means that it behaves in the same way as a subroutines call with the \(L\) word in the main NC program.

\section*{Notice}

This mechanism is primarily intended for initialisation, definition and presetting of offsets, G functions, parameter values, variables etc. Concrete machining processes should not be programmed here.

\subsection*{8.5 Implicit global subroutine call at program end}

The same recurring actions (e.g. deselecting transformations, offsets, functions etc.) must often be executed at the end (M02, M30) of different NC main programs. If these initialisations are summarised in a global subroutine, this program can be executed implicitly with the channel parameter P-CHAN-00252 before M02 or M30. before M02 or M30.

The full scope of NC programming functions can be used in this global subroutine. This means that it behaves in the same way as a subroutines call with the \(L\) word in the main NC program.

\section*{Notice}

This mechanism is primarily intended for initialisation, definition and presetting of offsets, G functions, parameter values, variables etc. Specific machining processes should not be programmed here.

\subsection*{8.6 Cycles as global or local subroutines (Call L | LL CYCLE)}

Cycles are available in the NC kernel in the form of global or local subroutines and permit special machining operations such as deep hole drilling or pocket milling. The machining task defined in the cycle is described in general form. When the cycle is invoked, the data is supplied when the transfer parameters are assigned.

A cycle is programmed independently of the currently valid plane (G17, G18, G19) and independently of the axis names configured in the NC channel. Only the direction from which machining should be executed in the current plane must be specified at cycle call. In the cycle, access can be made to an encapsulated group of parameters of its own. They are assigned values at cycle call.

A special syntax characterised by the "@" character is available for this purpose. This character is used in cycle programming in combination with:
@Pxx
@X, @Y, @Z
@I, @J, @K
@S

Transfer parameter in the cycle call and the cycle
Main axes in the cycle
Centre point coordinates in the cycle
Main spindle in the cycle

The cycle call must be programmed in its own NC block without any further NC commands. The syntax consists of a global or local subroutine call with additional specification of cycle-dependent transfer parameters.

L | LL CYCLE [ NAME=<cycle> [MODAL_MOVE / MODAL_BLOCK] @P1=.. @P200=.. \{ \\\(] }\)
\begin{tabular}{|c|c|}
\hline NAME=<cycle> & Name of cycle (file name) \\
\hline \begin{tabular}{l}
MODAL_MOVE \\
(MODAL, old syntax)
\end{tabular} & Modal cycle call. The cycle is again executed implicitly after every further NC block in the main program containing motion commands. \\
\hline \multirow[t]{6}{*}{MODAL_BLOCK} & Modal cycle call. The cycle is again executed implicitly after every further NC block in the active NC program. \\
\hline & \begin{tabular}{l}
With the following NC commands, the implicit block-modal cycle call is suppressed: \\
- Blank lines, comment lines
\end{tabular} \\
\hline & - Subroutine calls (L, LL, M6, G8xx). \\
\hline & - \$ commands (\$GOTO, \$IF, \$FOR etc...) \\
\hline & - Program end M functions (M2, M30, M17, M29) \\
\hline & The modal effect of MODAL_MOVE / _BLOCK is deselected by the NC command \#DISABLE MODAL CYCLE. \\
\hline @P1=<expr>... & List of transfer parameters. \\
\hline ... @P200=<expr> & A maximum of 200 @Pxx parameters of type REAL or STRING (as of V3.3079.25) can be transferred. Write and read accesses are only allowed within a cycle. The @Pxx parameters can be assigned direct values, any variables, P parameters and mathematical expressions. \\
\hline 1 & Separator ("Backslash") for programming the command over multiple lines \\
\hline
\end{tabular}

\section*{Transfer parameters - @P parameter}
- Inside the brackets, no order is required to specify the key words and supply parameters. When programming, users only need to know which @P parameter must be assigned for the cycle.
- @P parameters that are not needed can be omitted.
- With read access to a @P parameter that was not transferred to the cycle, it is implicitly created (default) and initialised with 0 (zero). This then increases the memory requirement. The user can switch off this behaviour using P-CHAN-00463 as of CNC Build V3.1.3079.20. When a read access is executed to @P parameters that are not transferred, the error ID 20394 is output. In CNC Builds up to V3.1.3079.19, non-programmed @P parameters are initialised with 0 (zero) on read access.
- The variable V.G.@P[i].VALID determines whether a @P parameter is used or is valid in the cycle.
- The functions IS_STRING and IS_NUMBER can be used to check whether @P parameters in the cycle are a string or a number, see Example 4 [ 215]. (as of V3.3079.25)
- In CNC Builds up to V3.1.3079.19, the transfer parameters are retained until the programmed call of another cycle (L CYCLE.. or G80.. [ 123]). As of CNC Build V3.1.3079.20, the transfer parameters are deleted when the cycle is closed (M17 or M29).
- @P parameters that are programmed in the cycle call but are not used in the cycle itself are ignored.
- The variable V.G.CYCLE_ACTIVE [> 584] determines whether the current subroutine or the current program level is a cycle.

\section*{Attention}

A cycle is a self-contained NC program unit with a defined machining task. It is advisable to avoid nested calls of cycles because there is a danger of assigning transfer parameters several times.

\section*{Notice}

Depending on the version, the processing lines of the cycle are masked or visible in the default setting in the running NC program or in single-block mode during the execution of a cycle in the display. When the display is off, only the cycle call is displayed during this time.
This feature is switchable by the channel parameter P-CHAN-00211.

\section*{Notice}

In TwinCat systems, all cycle lines are visible in the display by default.

\section*{Definitions required before cycle call:}
- Modal G functions, circle geometry data and the currently active feedrate ( F word) active before cycle call are retained beyond the cycle. This feature is switchable by the channel parameter P-CHAN-00210.
- Modal G functions that are programmed in combination with axis names (e.g. G92, G98, G99, G100, G112, G130, etc.) are not restored if they were programmed in the cycle itself.
- The machining plane (G17, G18, G19) should be defined in the higher-level NC program before cycle call. The axis perpendicular to this plane in drilling cycles is the axis in which the drilling operation is executed and in milling cycles it is the feed axis for depth.
- Any tool geometry compensation (e.g. length compensation) must also be selected before the cycle is invoked.
- The values required for feedrate, spindle speed and spindle rotation direction must be defined in the higher-level NC program, unless there are corresponding transfer parameters in the cycle.
- Spindle commands programmed in the cycles always refer to the active main spindle of the NC channel. Make sure that this main spindle is defined before cycle call.
- The start position for a corresponding drilling or milling operation and the tool's orientation must always be approached before the cycle is invoked in the higher-level NC program.

\section*{Deselecting a modal cycle}

A modal acting cycle (keyword MODAL_MOVE or MODAL_BLOCK in the cycle call) is deselected with the following NC command. The command must be programmed on its own in the NC block.

\section*{\#DISABLE MODAL CYCLE}

\section*{Available cycles:}

The following cycles are available:
- Machining cycles
- Calibration and measurement cycles
- and Cycles for kinematic optimisation Programing Example

The example below of a cycle call for drilling (drill.cyc) presents various parameter assignment parameter.

The drill.cyc drilling cycle requires the following transfer parameters:
@P1 Position of the retraction plane (absolute)
@P2 Position of the machining plane (absolute)
@P3 Safety distance (unsigned)
@P4 Final drilling depth (absolute) or
@P5 Final drilling depth relative to the machining plane (unsigned)

\section*{Cycle call with constant values:}
```

Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P4=40]
or by specifying a relative drilling depth @P5:
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P5=60]

```

\section*{Cycle call with variables:}
```

Variables must be defined and assigned values before cycle call.
..
\#VAR
V.L.RPL = 110
V.L.WPL = 100
V.L.SDST = 4
V.L.DEP = 50
\#ENDVAR
NxX L CYCLE [NAME=drilling.cyc @P1= V.L.RPL @P2=V.L.WPL @P3=V.L.SDST @P4=V.P.DEP]

```

Cycle call with P parameters:
```

The parameters must be defined and assigned values before cycle call.
Nxx P10 = 110

```
```

Nxx P11 = 100
Nxx P15 = 4
Nxx P17 = 50
Nxx L CYCLE [NAME=drilling.cyc @P1= P10 @P2=P11 @P3=P15 @P4=P17]

```

\section*{Cycle call with mathematical expressions:}
```

Nxx P20=100

```
Nxx L CYCLE [NAME=drilling. cyc @P1=10+P20 @P2=2*50 @P3=5-1 @P4=P20/2]

\section*{Cycle call with constant; any sequence of parameters in brackets:}
```

NXX L CYCLE [@P4=40 NAME=drilling.cyc @P2=100 @P3=4 @P1=110]

```

\section*{Cycle call with constant values; cycle should have a modal effect:}
```

Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P4=40 MODAL_MOVE]

```
[Example:]
\%drill main
N05 T1 D1
N10 M06
N15 G53 G17 G90 M3 S300 F200 S300
N16 G0 X0 Y0 Z0
N20 Z110
N30 X40 Y40 (drill position 1)
N40 L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=2 @P4=55 MODAL MOVE]
N50 X60 Y60 (drill position 2 and implicit cycle call because it is modal)
N60 X100 Y60 (drill position 3 and implicit cycle call because it is modal)
N70 X100 Y20 (drill position 4 and implicit cycle call because it is modal)
\#DISABLE MODAL CYCLE
N80 X0 Y0 M5
N100 M30

\section*{Notice}

\section*{Notes on creating cycles}

As far as possible, cycles must be programmed as generally valid and independently of the axis names currently used in the NC channel and the definition of planes. For this purpose, the cycle has the option of using plane-independent "neutral axis names " @X, @Y and @Z for the first three main axes. Meanings:
@X always the first main axis
@Y always the second main axis
@Z always the third main axis

Example 1: axes in the cycle
```

Nxx G91 @X=@P1 @Y=@P2 @Z=@P3 F1000 G01

```

By analogy, so-called "neutral centre point coordinates" are available for programming circles.
Meanings:
@l always the centre point coordinate in the first main axis
@J always the centre point coordinate in the second main axis
@K always the centre point coordinate in the second main axis

\section*{Example 2: circle in the cycle}

Nxx G91 G02 @X=@P1 @Y=@P2 @I=@P4 @J=@P5 F1000

To remain independent from the spindle name configured during spindle programming, the main spindle can always be addressed in the cycle by the neutral spindle name @S.
@S always the main spindle

Example 3: spindle in the cycle

Nxx @S=1000 M3 (main spindle cw at 1000 rpm)

\section*{Example 4: Check @P parameter}

Transferred @P parameters can be checked by the functions IS_STRING and IS_NUMBER.
```

%L cycle
( Check variables)
\$IF IS STRING[@P1] == TRUE
\#MSG["Text: %s",@P1]
\$ELSE
\#MSG["Error no String"]
\$ENDIF
\$IF IS_NUMBER[@P2] == TRUE
\#MSG["N
\$ELSE
\#MSG["Error not a number"]
\$ENDIF
M17
% Main
LL CYCLE [NAME=cycle @P1 ="String1" @P2=12.34]
M30

```

\subsection*{8.7 Calling block sequences (L SEQUENCE)}

Block sequences are contiguous program parts or single NC blocks in the current NC program or a global subroutine which can be executed once or several times with L SEQUENCE.

A block sequence is defined by specifying the start and end identifications by:
- Block numbers N.. or
- Jump labels ([Stringlabel] analogous to the definition for \$GOTO)

\section*{Notice}

Every call of a block sequence is identical to a subroutine call. The same rules on nesting depth apply as for global subroutines.

\section*{Attention}

\section*{Context evaluation:}

The program context in the subroutine is not set up until the first NC line of the block sequence is executed. All previous NC lines passed through are not evaluated. Previously defined variables/ coordinate systems, parameters, modal statements, etc. are neither created nor initialised. Therefore, they are unknown or not available in the block sequence.

In particular when executing block sequences with control block statements (\$IF-\$ELSE-\$ENDIF, \$SWITCH,..), users must ensure themselves that they pass through the entry and return points without conflict.

Syntax of L SEQUENCE when using block numbers:

L SEQUENCE [ [ NAME=<string>] N<expr> [ N <expr> ] [REPEAT<expr> ] [ ENDTAG ]]
\begin{tabular}{ll} 
NAME=<string> & \begin{tabular}{l} 
Name of the current subroutine or a global subroutine in which the block sequence \\
makes a pass. Optional: If no name is programmed, the block sequence passes \\
through the current NC program.
\end{tabular} \\
\(\mathrm{N}<\) expr \(>\) & \begin{tabular}{l} 
Number of the first block to be executed (start number, start of block sequence)
\end{tabular} \\
\(\mathrm{N}<\) expr \(>\) & \begin{tabular}{l} 
Number of the last block to be executed (return number, end of block sequence) Op- \\
tionally, if both block numbers are identical, only this block is executed.
\end{tabular} \\
REPEAT<expr> & \begin{tabular}{l} 
Number of repetitions of a block sequence, positive integer \(\geq 1\). Optionally, if REPEAT \\
is not specified, the block sequence makes a single pass.
\end{tabular} \\
ENDTAG & \begin{tabular}{l} 
Marks the call L SEQUENCE itself as an additional valid end of the block sequence. \\
Optionally, if both ENDTAG and a return number N.. are programmed at the same \\
time, the sequence end found first is the valid one.
\end{tabular}
\end{tabular}

The controller searches for the programmed N (block) numbers in the specified NC program (which can also be the same program that calls up the command). The two N numbers mark the first and last NC blocks to be executed in the block sequence; NC blocks outside this block sequence are not executed.
It is recommended to use a unique ascending numbering format for the NC blocks.

The start and return numbers can also be swapped in the command when programmed. In the NC program, however, the block sequence always passes through from the lower N number to the higher N number.

An error message is issued if the start or return number is not found.
If the block sequence is to be executed multiple times (REPEAT > 1), the program starts at the start number again at the end of the block sequence. Once all passes have been executed, the program returns from the block sequence and the rest of the program sequence is continued.

If only an N number was specified in the command, only this line is passed. This corresponds to a call with two identical N numbers.

The L SEQUENCE call itself may be located in the block sequence defined by the N numbers. When the same call is read again, the following two reactions are possible:

Without ENDTAG the recall is ignored and the block sequence is executed until the return number.

With ENDTAG the L SEQUENCE call is marked as the valid sequence end and the block sequence is terminated.

\section*{Programing Example}

\section*{Calling block sequences (L SEQUENCE)}

Repeat block sequence between 2 block numbers once:
```

N20 ... ;Start number
N50 ... ;Return number
N80 L SEQUENCE [N20 N50] ; or..
N80 L SEQUENCE [N20 N50 REPEAT=1]

```

Repeat block sequence between 2 block numbers several times:
```

N20 ... ;Start number
N50 ... ;Return number
N80 L SEQUENCE [N20 N50 REPEAT=4]

```

Block sequence between 2 block numbers. Bracketed sequence call is ignored when the block sequence is executed since ENDTAG is not set:
```

N20 ... ;Start number
N40 L SEQUENCE [N2O N80]
N80 ... ;Return number

```

Block sequence between 2 block numbers. Bracketed sequence call is the first block sequence end found since ENDTAG is set:
```

N20 ... ;Start number
N40 L SEQUENCE [N20 N80 ENDTAG]
N80 ... ;Return number

```

Block sequence between 2 block numbers with ENDTAG. ENDTAG is not relevant since the return number is before the sequence call:
```

N20 ... ;Start number
N50 ... ;Return number
N80 L SEQUENCE [N20 N50 REPEAT=4 ENDTAG]

```

Repeat a single NC block several times:
```

N20 ... ;Start number
N80 L SEQUENCE [N20 REPEAT=4] ; or..
N80 L SEQUENCE [N20 N20 REPEAT=4]

```

Block sequence between 1 block number and ENDTAG:
```

N20 ... ;Start number
N80 L SEQUENCE [N20 ENDTAG]

```

Repeat block sequence between 2 block numbers several times. Sequence call is before the block sequence:
```

N80 L SEQUENCE [N100 N150 REPEAT=4]
N100 ... ;Start number
N150 ... ;Return number

```

Nested multiple call of block sequences between block numbers:
```

N40 L SEQUENCE [N60 N150 REPEAT=2] ; Sequence call 1
N60 ... ;Start number 1
N90 ... ;Start number 2
N120 ... ;Return number 2
N130 L SEQUENCE [N90 N120 REPEAT=4] ;Sequence call 2
N150 ... ;Return number 1

```

Repeat block sequence between 2 block numbers in a global subroutine several times:
```

N20 ...
N80 L SEQUENCE [NAME="glob_1.nc" N50 N150 REPEAT=4]

```

Nested multiple calls of block sequences in the current program and a global subroutine between block numbers:
```

N20 L SEQUENCE [N60 N150 REPEAT=2] ; Sequence call 1
N60 ... ;Start number 1
N80 L SEQUENCE [NAME="glob_1.nc" N50 N150 REPEAT=3] ;Sequence call 2
N150 ... ;Return number 1

```

\section*{Syntax of L SEQUENCE when using jump labels (string labels):}

L SEQUENCE [ [ NAME=<string>] [<START>] [ [<END>]] [ REPEAT<expr>] [ ENDTAG ]]
\begin{tabular}{ll} 
NAME \(=<\) string \(>\) & \begin{tabular}{l} 
Name of the current subroutine or a global subroutine in which the block sequence \\
makes a pass. Optional: If no name is programmed, the block sequence passes \\
through the current NC program.
\end{tabular} \\
{\([<S T A R T>]\)} & \begin{tabular}{l} 
Start label of first block to be executed (start of block sequence) \\
{\([<E N D>]\)} \\
End label of last block to be executed (return, end of block sequence) Optionally, if \\
both labels are identical, only this block is executed.
\end{tabular} \\
ENDEAT<expr> & \begin{tabular}{l} 
Number of repetitions of a block sequence, positive integer \(\geq 1\). Optionally, if REPEAT \\
is not specified, the block sequence makes a single pass.
\end{tabular} \\
& \begin{tabular}{l} 
Marks the call LSEQUENCE itself as an additional valid end of the block sequence. \\
Optionally, if both ENDTAG and an end label are programmed at the same time, the \\
sequence end found first is the valid one.
\end{tabular}
\end{tabular}

The controller searches for the programmed jump labels in the specified NC program (which can also be the same program that calls the command). The two jump labels mark the first and last NC blocks to be executed in the block sequence; - NC blocks outside this block sequence are not executed

Jump labels are set at block start or directly after the block number. An error message is issued if the start or return label is not found.

If the block sequence is to be executed multiple times (REPEAT \(>1\) ), the program starts at the start label again when it reaches the end of the block sequence. Once all passes have been executed, the program returns from the block sequence and the rest of the program sequence is continued.

If only a single start label was specified in the command, only this line is passed. This corresponds to a call with two identical N numbers.
The L SEQUENCE call itself may be located in the block sequence defined by the jump labels. When the same call is read again, the following two reactions are possible:

Without ENDTAG the recall is ignored and the block sequence is executed up to the return label.
With ENDTAG the L SEQUENCE call is marked as the valid sequence end and the block sequence is terminated.

\section*{Programing Example}

Calling block sequences (L SEQUENCE)

Repeat block sequence between 2 jump labels once:
```

N20 [STARTLBL] ... ; Start label
N50 [ENDLBL] ... ;Return label
..
N80 L SEQUENCE [[STARTLBL] [ENDLBL]] ;oder..
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=1]

```

Repeat block sequence between 2 jump labels several times:
```

N20 [STARTLBL] ... ;Start label
N50 [ENDLBL] ... ;Return label
N80
L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4]

```

Block sequence between 2 jump labels. Bracketed sequence call is ignored when the block sequence is executed since ENDTAG is not set:

N20 [STARTLBL] ... ;Start label
N40 L SEQUENCE [[STARTLBL] [ENDLBL]]
...
N80 [ENDLBL]...
;Return label

Block sequence between 2 jump labels. Bracketed sequence call is the first block sequence end found since ENDTAG is set:
```

N20 [STARTLBL]... ;Start label
N40 L SEQUENCE [[STARTLBL] [ENDLBL] ENDTAG]
N80 [ENDLBL] ... ;Return label

```

Block sequence between 2 jump labels with ENDTAG. ENDTAG is not relevant since the return label is before the sequence call:
```

N20 [STARTLBL]... ; Start label
N50 [ENDLBL] ... ;Return label
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4 ENDTAG]

```

Repeat a single NC block several times:
```

N20 [STARTLBL] ... ;Start label
N80 L SEQUENCE [[STARTLBL] REPEAT=4] ;or..
N80 L SEQUENCE [[STARTLBL] [STARTLBL] REPEAT=4]

```

Block sequence between 1 jump label and ENDTAG:
```

N20 [STARTLBL] ... ;Start label
N80 L SEQUENCE [[STARTLBL] ENDTAG]

```

Repeat block sequence between 2 jump labels several times. Sequence call is before the block sequence:
```

N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4]
N100 [STARTLBL] ... ;Start label
N150 [ENDLBL] ... ;Return label

```

Nested multiple call of block sequences between jump labels:
```

N40 L SEQUENCE [[STARTLBL1] [ENDLBL1] REPEAT=2] ;Sequence call 1
N60 [STARTLBL1] ... ;Start label 1
N90 [STARTLBL2] ... ;Start label 2
N120 [ENDLBL2] ... ;Return label 2
N130 L SEQUENCE [[STARTLBL2] [ENDLBL2] REPEAT=4] ;Sequence call 2
N150 [ENDLBL1]... ;Return label 1

```

Repeat block sequence between 2 jump labels in a global subroutine several times:
```

N20 ...
N80 L SEQUENCE [NAME="glob_1.nc" [SUP1] [EUP1] REPEAT=4]

```

Nested multiple calls of block sequences in the current program and a global subroutine between jump labels:
```

N20 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=2] ;Sequence call 1
N60 [STARTLBL] ... ;Start label 1
N80 L SEQUENCE [NAME="glob_1.nc" [SUP1] [EUP1] REPEAT=3] ; Sequence call 2
N150 [ENDLBL]... ;Return label 1

```

\section*{\(9 \quad\) Parameters and parameter calculation (P)}

In NC programs, parameters can be used as placeholders for numerical values. The advantage of parameters is that the value of a parameter may be changed during the program flow. This allows the production of flexible NC programs.

A parameter is designated by " P " followed by a number without blank.

\section*{Programing Example}

\section*{Parameters and parameter calculation}

In a sub-routine, e.g. a drilling cycle, instead of coordinate values (drill depth, drill feed, dwell etc.) parameters are used. The parameters are then assigned the final values in each calling main program:

For the global sub-routine
\%4712(drilling, face countersinking)
the following parameters are to be defined:
P10 -Reference plane=withdrawal plane
P11-Drilling depth
P12-Dwell period


Fig. 64: Application example of parameter calculation

The call in the main program then looks like this:
```

N100 P10=20.5 P11=12.6 P12=1.2
N110 L4712

```
:
\(\mathbf{P}\) <expr> simple parameter
\(\mathrm{P}<e x p r>\quad\) The parameter index must always be greater than zero. However, it can assume any desired value. The maximum number of parameters used in the channel is fixed [6] [ 819]-6.19.
Parameter arrays (e.g. P100[50]) are also allowed in addition to plain parameters. The dimension of the arrays is fixed [6] [> 819]-6.20.

\section*{\(\mathbf{P}<e x p r>[<e x p r>]\) \{ [<expr>] \}}

The channel parameter P-CHAN-00067 specifies whether the \(P\) parameters are active program global.

Parameters can be created in the NC program in a declaration block (and initialised as required).
It starts with \#VAR and ends with \#ENDVAR Or implicitly at the first write access. However, parameter arrays must always be created in a declaration block.

For a better overview, the initialisation of a parameter array can be written over several NC blocks by using the "l" character. The following syntax is used for the creation operation:
```

\#VAR Start of declaration block
:
:
\#ENDVAR
End of declaration block

```
```

\#VAR
P10[3][6] = [10,11,12,13,14,15, \
20,21,22,23,24,25, \
30,31,32,33,34,35 ]
P20[3][4] = [40,41,42,43, 50,51,52,53, 60,61,62,63]
P100
\#ENDVAR
P200 = 10 P201=11
:

```

\section*{Notice}

Access to parameter arrays starts at index 0 . Based on the example above, access \(\mathrm{P} 10[0][5]\) gives the value 15 .

Parameters and parameter arrays can also be deleted in the NC program. The \#DELETE command with the following syntax is provided for this:
```

\#DELETE P<expr> {, P<expr>}

```

\section*{Programing Example}
\#DELETE P
\#DELETE P10, P20, P100, P200, P201

In addition, the SIZEOF and EXIST functions are provided (see Section Arithmetic expressions <expr> [ \({ }^{21}\) ]) to define the dimension size of parameter arrays and to check the existence of parameters.

Parameters receive their values assigned by the NC program, e.g. P12=0.12. They also allow the processing of control-dependent or process-dependent values of the control system, e.g.:
- current spindle rpm,
- torque in the drives,
- values of external measuring devices
- values of heat or force sensors
- keyboard inputs via operating menu
etc.

Linked arithmetical expressions can also be used instead of the direct assignment of numerals (see Section Mathematical expressions [ 30]). The known mathematical rules apply for inputs, e.g.:
- point-before-slash calculation,
- The parentheses rule; however, the square parentheses "[ ]" must be used.

\subsection*{9.1 Programming of coordinates by parameters}

The syntax when programming coordinates for axis designations is:
```

<axis_name> P<expr>

```
<axis_name>
P<expr>

Designation of the axis
Assigned parameter
\(\mathrm{P}<e \operatorname{expr}>\mathrm{P}<e x p r>\) can also be formed by a mathematical expression.

\section*{Programing Example}

Programming of coordinates by parameters

X P1*SIN [P2*30]

\subsection*{9.2 Indirect parameters}

In arithmetical expressions and assignments, indirect parameters are used in the same way as direct parameters.

Both direct (Pnn) and indirect programming (PPnn) is performed using the P word. When indirect parameters are used, the following applies:

PPnn points to the parameters Pnn.

When a PPnn is initialised, the address of a Pnn is assigned. The use of PPP... is also possible.

If \(P 120=10\), the value 10 is loaded to the parameter 120. However, the statement PP120 \(=123.456\) assigns this value to the parameter whose address exists in P120, i.e. P10. Accordingly, PP121=SQRT[2,0] produces the following result:


Fig. 65: Illustration of the effect of indirect \(P\) parameters

The use of indirect parameters permits the assignment of entire fields of parameters:

\section*{Programing Example}

\section*{Indirect parameters}
```

Assignment of P parameters P20 and P40 with 50
:
N110 P1 = 20 P2 = 40
N120 PP1 = 50
N130 PP2 = PP1
Assignment of P parameters P15 to P25 with 0.0
N110 \$FOR P1 = 10,20,1
N120 P[P1 + 5] = 0.0
N130 \$ENDFOR

```
kernel Industrielle Steuerungstechnik GmbH

\section*{10 Statements for influencing NC program flow}

A complete list of \(G\) functions is contained in the overview of commands in the Appendix under Control block statements (\$..) [〉 808].

The syntax for control block statements is:

\section*{\$<statement>}

Control block strings as described in Section Conditional jumps [ \(>232\) ]. Note that between \$ and <statement> no blank characters are permissible.

Statements for influencing program flow (control blocks) permit the implementation of:
- Conditional jumps, e.g. to trigger optional machining steps depending on a measured value
- Incremental digital loops to simplify the programming of several repetitive machining steps, e.g. for line milling or for drilling hole circles
- Loops with running condition to allow the repetition of several machining steps until the abort condition is fulfilled. For example, if the infeed of the tool and a machining operation are to be carried out until a definite coordinate value is reached. Loops may be programmed as endless loops if a running condition is missing or not fulfilled.

The following rules apply for the use of control blocks:
- Only one control block may be present in one NC block.
- Control block statements may be nested. The nesting depth is fixed.
- Only the block number and "/" may be programmed in front of the control block.
- In the invalid branch of a control block statement, a syntax check is performed for block numbers and other (nested) control block statements (see examples of IF-ELSE branching).

\section*{Programing Example}

\section*{Syntax check in an invalid branch:}
```

N10 \$IF 0
N2O XY
N30 \$ENDIF
N10 \$IF 0
N20 ...
N30 \$IF XY
(Syntax check due to nested control block)
(statement; error message due to unknown term)
N40 ...
N50 \$ENDIF
N60 \$ENDIF
N10 \$IF 0
NXY
N30 \$ENDIF
(Here no syntax check takes place)
...
(Syntax check of block number;)
(statement; error message due to unknown term)
Due to inaccuracies in the calculation and the internal representation of parameters, comparative operations (see Section Arithmetic expressions <expr> [> 31]) in control block statements may lead to an erroneous result. Therefore in cases of doubt, check for a tolerance range instead of for precise values.

```

\section*{Programing Example}

\section*{WRONG:}

N10 \$FOR P1 = 0, 10, 1
N20 P2 = P2 + 0.01
N30 \$ENDFOR
N40 \$IF P2 == 0.1 (Due to inaccuracies of calculation P2)
N50 ...
(may be unequal to 0.1 so that the \$ELSE)
N60 \$ELSE
(branch is executed)
N70 G04 X20
N80 \$ENDIF

\section*{RIGHT:}

N10 \$FOR P1 = 0, 10, 1
N20 P2 = P2 + 0.01
N30 \$ENDFOR
N40 \$IF ABS[P1 - 0.1] <= . 000001 (Check a tolerance range for)
N50 G04 X5 (unproblematic NC machining)
N60 \$ELSE (\$IF branch is)
N70 ...
(executed)
N80 \$ENDIF

\subsection*{10.1 Conditional jumps}

\subsection*{10.1.1 The IF - ELSE branch}

The following control statements are used for IF-ELSE branches:
\$IF, \$ELSE, \$ELSEIF, \$ENDIF.

Branching always starts with
\$IF <expr>
and always ends with
\$ENDIF
Control statements
\$ELSE
and
\$ELSEIF
are optional and serve to set up multiple branches.

\section*{Attention}

The condition in the \$IF control block is checked by verifying the mathematical expression for "true" or "not true" (TRUE and FALSE). To be able to also use decimal variables, the jump condition is regarded as fulfilled (TRUE) if...
...the absolute value of the mathematical expression is \(\mathbf{>}\) or \(=0.5\).

\section*{Programing Example}

The IF - ELSE branch
```

N10 ...
N20 \$IF P1 Only if |P1| is greater or equal to 0.5 are the
statements N30 to N50 executed.
N30 ...
N40
N50
N60 \$ENDIF

```

However, the following is also possible :
N10 ...
N20 \$IF P1 >= 0.5 Only if P1 is greater or equal to 0.5 are the statements N30 to N50 executed.
N30 ...
N40
N50
N60 \$ENDIF
or:
N10 ...
N20 \$IF P1 > P2
Only if P1 is greater than \(P 2\) are the statements N30 to N50 executed, otherwise N70 to N90
N30 ...
N40 ...
N50 ...
N60 \$ELSE
N70 ...
N80 ...
N90 ...
N100 \$ENDIF

\section*{These use of ELSEIF permits:}

N10 ...
N20 \$IF P1 == Only if P1 is equal to 0 are the statements N30 to N50 executed, otherwise a check is made in the \$ELSEIF condition whether P2 is >= 0.5 and accordingly N70 to N90 or N110 to N130 are executed.
N30 ...
N40
N50
N60 \$ELSEIF P2>=0.5 The \$ELSEIF conditions is used to form nested branches.
N70 ...
N80
N90
N100 \$ELSE
N110 ...
N120
N130
N140 \$ENDIF

\section*{Attention}

The C programming language also makes a distinction in syntax between
Assignment : P5 = 3
and
Comparison: \$IF P5 == 3
The following applies to Version 2.3 and earlier: As mathematical expressions expect always the sequence...
Operator -> Term -> Operator -> Term -> etc.
...expressions preceded by a minus sign must be bracketed in comparison operation ("-" is interpreted as operator).

\section*{Programing Example}
\begin{tabular}{ll} 
\$IF P1 >= -5 & incorrect since the term->operator->operator->term \\
\$IF P1 >= [-5] & correct since term->operator->term->operator->term
\end{tabular}

\subsection*{10.1.2 Switch branching (\$SWITCH )}

SWITCH branching permits the processing of various NC program variants as a function of an arithmetic expression.
The control statements given below are used
\$SWITCH, \$CASE, \$DEFAULT, \$ENDSWITCH
for branching.
Branching always starts with:
\$SWITCH <expr1>
followed by several
\$CASE <expr2> ...
\$BREAK
optionally followed by:
\$DEFAULT
and always ends with

\section*{\$ENDSWITCH}

\section*{Programing Example}

SWITCH branching
```

N100 \$SWITCH P1=INT [P1*P2/P3] If the result of the arithmetic expression
is equal to 1, the blocks after \$CASE 1 are
N110 \$CASE 1 executed (N120-140)
N120 ...
N130
N140 \$BREAK
N150 \$CASE P2 If the result is equal to P2, the blocks
N160..N170 are executed.
N160 ...
N170 \$BREAK
N300 \$CASE n
N320 ...
N330 \$BREAK
N350 \$DEFAULT The \$DEFAULT block is optional and serves to
process the NC blocks N360-N380.
N360 ... if the result of the \$SWITCH block
N370 does not match any of the \$CASE cases.
N380
N390 \$ENDSWITCH

```

\section*{Notice}

i
The expression <expr1> and <expr2> are compared using the internal REAL mode. Here, both expressions are evaluated as equal if the value difference is \(<0.001\).
The expressions <expr1> and <expr2> can also assume negative values.

\subsection*{10.1.3 The \$GOTO statement}

In addition to the subroutine technique or the use of control block statements (\$IF, \$FOR...), this functionality offers another option for branching to other program parts. The GOTO command can be called at any point in the NC program by setting jump labels in the NC program.

There are two options to use jump statements:
Expression - Label:
\begin{tabular}{|ll|}
\hline N<block_number>: & Definition \\
\hline \$GOTO N<Block_number> | \$GOTO N <expr> & Jump call \\
\hline
\end{tabular}

\section*{String - Label:}
[<string>] Definition
\$GOTO [<string>] Jump call

\section*{Characteristics:}
- The \$GOTO call can be placed in the NC program before or after the label definition. A label search is carried out in the program in the upwards and downwards directions.
- The label must always be called and defined at the same program level (locally within the program). Program jumps between the main program and a subroutine as well as jumps between subroutines are not permitted (see figure below).
- Identical labels may be defined in main programs and subroutines.
- It is possible to jump to the same label from several points in the NC program.
- A \$IF statement may be combined with a \$GOTO in the same NC line. In this case, no associ-

- Other NC commands may be programmed before and after a \$GOTO command in the same NC line. However, the jump is the last action in the NC line.
- External jumps are possible to any levels of a \$IF-\$ELSE-\$ENDIF control block and within and between these levels. Then however, this jump-in level is the active level (condition is assumed as true; see Programming example).
- Jumps within \$WHILE, \$FOR, \$DO, \$REPEAT are not permitted.
- Complete exit from any control block statement by a \$GOTO from any level is always permitted.
- Labels in comments (\#COMMENT BEGIN, \#COMMENT END) are not recognised.
- In string labels, no distinction is made between uppercase and lowercase.
- All labels ignored during decoding are stored. The maximum number of storable expression labels [6] [> 819]-6.41, string label [6] [> 819]-6.42 and string label length [6] [> 819]-6.43 are specified.
- At each jump call, a check is made whether the jump label already is known, i.e. stored. If the check is positive, the jump is executed immediately. If the jump label is unknown, a search is made from the current NC line at program level up to program end (M29/M30). If the jump label is not found, error message P-ERR-20840 is output.
- After reaching the maximum number of storable labels and other new jump labels are decoded, these labels are no longer saved. This is displayed by the warnings P-ERR-20829 or P-ERR-20831. For every new jump label call with an unknown jump label the search starts again at the beginning of the current program level In this case, the jump process may require more time with very large NC programs.

\section*{Programing Example}

The \$GOTO statement
```

%goto
N05 P1=1
N06 P2=1
N10 G74 X1 Y2 Z3
N11 X0 Y0 Z0
N15 \$IF P1==1 \$GOTO N4O: ; Jump from outside to N40 in a
; control block
N20 X10
N25 Y10
N30 \$IF P1==2
N35 X20
N40: \$IF P2==1
N45 X30
N50: Y30 \$GOTO N65: ; Jump to N65 between control block levels
\$ENDIF
N55 \$ELSE
N60 Y40
N65: X40
N70 \$ENDIF
N80 Z99
N999 M30

```


Fig. 66: Permitted and impermissible jumps in the \$GOTO command

\section*{Programing Example}
```

N10 G1 XY
N20: X100 ;Label definition N20:
\$IF V.L.dummy 1 <100 \$GOTO [N20]
\$IF V.L.dummy 1 >200
\$GOTO [LABEL_1
Y20
\$ENDIF
[LABEL_1] X0 ;Label definition [LABEL_1]
N30 AO
\$FOR V.P.my-var = 0, 4, 1
\$IF V.L.dummy_2 <200 \$GOTO [CONTINUE] ; Jump to label [CONTINUE])
\$SWITCH V.P.my-var
\$CASE 0
V.P.AXE-X=V.P.GROUP[1].position[V.P.my-var]
\$BREAK
\$CASE 1
V.P.AXE-Y=V.P.GROUP[1].position[V.P.my-var]
\$BREAK
\$CASE 2
V.P.AXE-Z=V.P.GROUP[1].position[V.P.my-var]
\$BREAK
\$CASE 3
V.P.AXE-A=V.P.GROUP[1].position[V.P.my-var]
\$DEFAULT
\$ENDSWITCH
\$ENDFOR
[CONTINUE] ;Label definition [CONTINUE]
N1000 ...

```

\subsection*{10.1.3.1 Parametric jump call}

The \$GOTO command can also program the jump label destinations in parametric form. This permits the external control of an NC program flow (e.g. from the PLC).
In the case of a jump call for expression labels, all the mathematical expressions provided <expr> in the syntax scope to display block numbers are permissible, e.g. parameters, local and global variable and external variables.

\section*{\$GOTO N<expr> Jump call}

The jump call of string labels can be parameterised by external variables of the string or string array type (see also Section External variables (V.E.) [ 615]). The name of the jump label is then stored in the external variable.
\$GOTO V.E. ...
Jump call

\section*{Programing Example}

\section*{Parametric jump call}
```

N10 ...
:
N50 \$GOTO NV.E.JUMP_EXPR ;(Jump e.g. to N200 via ext. Variable
;V.E.JUMP_EXPR containing the value 200
:
N100 \$GOTO V.E.JUMP_STR ;Jump e.g. to [CONTINUE) via ext. Variable
;(V.E.JUMP_STR containing CONTINUE string
:
:
N200:...
:
N500:...
:
:

```

```

[CONTINUE]..
:
N...

```

\subsection*{10.2 Counting loop (\$FOR)}

Counting loops permit the processing of statements \(n\) times. The number of loop passages is checked by a counting variable.

The syntax of a counting loop starts with:
```

\$FOR P<expr> = <expr1>, <expr2>, <expr3>
and always ends with:

```

\section*{\$ENDFOR}

Here, \(\mathrm{P}<e \operatorname{expr}>\) is the counting variable. Its start value is specified by <expr1>, its end value by <expr2> and the counting increment by <expr3>.

\section*{Notice}

Only integer values may be used as counting variables.
If decimal numbers are used, it is not possible to precisely represent the increment exactly (exception: powers of two) since a rounding error accumulates when added. This may lead to a loop that passes through one loop too few.

Instead of the P parameters, it is also possible to use variables ("V.") with write access.
If the counting increment is negative, the loop is aborted if the end value is undershot; if the counting increment is positive, the loop is aborted if the end value is exceeded. Programming the counting increment 0 leads to an endless loop and to the output of a warning.

\section*{Programing Example}

Counting loops
```

```
N100 $FOR P1= 10, 100, 2 P1 is pre-assigned the value 10 at loop start.
```

```
N100 $FOR P1= 10, 100, 2 P1 is pre-assigned the value 10 at loop start.
        The counting loop is passed until P1
        The counting loop is passed until P1
        exceeds the value 100; then P1 is
        exceeds the value 100; then P1 is
        incremented by 2 at the end of every loop
        incremented by 2 at the end of every loop
        pass.
        pass.
        Within the counting loop, the NC blocks N110
        Within the counting loop, the NC blocks N110
        to N130 are executed..
```

```
        to N130 are executed..
```

```

\section*{Programing Example}
```

Negative step width:
N100 \$FOR P1= 100, 10, -2 P1 is pre-assigned the value 100 at loop start.
The counting loop is passed until
N110 X SIN [P1 * 5] P1 undershoots the value 10; then P1 is
decremented by 2 at the end of every loop
pass. In the counting loops, the NC blocks N110 to
to N130 are executed..
N120 Y COS [P1 * 5]
N130 ...
N150 \$ENDFOR
Loops not executed:
N100 \$FOR P1= 100, 10, 1 P1 is pre-assigned the value 100 at loop start.
The counting loop is passed until
P1 exceeds the value 10.
N110 X SIN [P1 * 5] But here no loop since P1 is pre-assigned the
N120 Y COS [P1 * 5]
N130 ...
N150 \$ENDFOR
Endless loop:
N100 P2=20
N110 \$FOR P1= 100, 10, O Endless loop
N120 \$IF P2 == 50
N130 \$BREAK
N140 \$ENDIF
N150 \$ENDFOR

```

\subsection*{10.3 Loops with running condition}

\subsection*{10.3.1 Verification of running condition at loop start (\$WHILE)}

The syntax of this loop starts with:
\$WHILE <expr>
and always ends with

\section*{\$ENDWHILE}

At the start of every loop pass, the stated parameters are verified. The loop is aborted if the expression <expr> assumes the value range FALSE ( \(-0.5<\) expr < 0.5 ).

\section*{Programing Example}

\section*{Verification of running condition at loop start}
```

N90 P1 = 100.0
N100 \$WHILE P1 > 0.5 P1 > 0.5 is tested for FALSE at loop start
N110 P1 = P1 - 1.5 YP1 überprüft. The loop is passed until P1
N120 \$ENDWHILE
fulfills the abort condition
N130 ..

```

\subsection*{10.3.2 Verification of running condition at loop end (\$DO), (\$REPEAT) \\ There are two kinds of loops available. \\ The syntax of the DO loop starts with:}

\section*{\$DO}
and always ends with

\section*{\$ENDDO <expr>}

The stated parameters are checked at the end of every loop pass. The loop is aborted if the expression <expr> assumes the value range FALSE (expr < 0.5).

The syntax of the REPEAT loop starts with:

\section*{\$REPEAT}
and always ends with

\section*{\$UNTIL <expr>}

The stated parameters are checked at the end of every loop pass. The loop is aborted if the expression <expr> assumes the value range TRUE (expr> 0.5).

\section*{Notice}

As opposed to the \$WHILE and \$FOR loops, the \$DO and \$REPEAT loops are always passed at least once.

\section*{Programing Example}

Verification of running condition at loop end
```

N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 \$DO
N40 P1=P1+1
N50 XP1
N60 \$ENDDO P1 <= P2 P1 is checked for FALSE at loop end.
The loop is passed till P1 no longer fulfils the
condition.
N99 M30
N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 \$REPEAT
N40 P1=P1+1
N50 XP1
N60 \$UNTIL P1 > P2 P1 is checked for TRUE at loop end.
The loop is passed till P1 no longer
fulfills the condition.

```
N99 M30

\subsection*{10.4 Influencing loop flow sequences}
10.4.1 The \$BREAK statement

\section*{\$BREAK}

It is not always useful to exit a loop with the abort criterion. The keyword \$BREAK can also abruptly terminate the execution of a loop in addition to program execution with single \$CASE labels of the \$SWITCH statement (see Section SWITCH branching [〉 235]).

For example, this is useful with extremely nested loops if execution of the innermost loop should be interrupted.

Programing Example

The \$BREAK statement
```

N10 \$WHILE <expr1>
N20 ...
N30
loop is terminated if
expr1 "not valid" or
expr2 is "valid".
N40 \$IF <expr2>
N50 \$BREAK
N60 \$ENDIF
N70 ...
N80
N90 \$ENDWHILE
N100 ...

```

\subsection*{10.4.2 The \$CONTINUE statement}

\section*{\$CONTINUE}

As opposed to \$BREAK, the \$CONTINUE statement does not abort the loop but branches it to the loop start. All statements after \$CONTINUE are then not executed.

\section*{Programing Example}

The \$CONTINUE statement
```

N10 \$FOR <expr1>
N20 ...
N30
N40 \$IF <expr2>
N50 \$CONTINUE
N60 \$ENDIF
N70 ...
N80
N90 \$ENDFOR
N100 ...

```
```

The statements in the N7O and N80 lines

```
The statements in the N7O and N80 lines
are only executed if
are only executed if
expr2 is "not valid".
```

expr2 is "not valid".

```

\section*{11 Smoothing methods}

Introduction
A programmed curve must be rounded and smoothed within specific tolerances to allow it to move even over corners without stopping as quickly and uniformly as possible. This is referred to as smoothing and there are various methods provided.

Simple contours with few long linear and circular blocks are ideal for polynomial contouring. Select and this function with G261 and deselect with G260 . This method is described in Section G functions [> 127].

On the other hand, it is preferable to use \#HSCmethods for many short linear blocks. The methods include the highly rugged SURFACE method [ 253] that is particularly suited to free-form surface machining. It achieves the best results in the event of disruptions in the programmed contour and blocks which have very different lengths. By contrast, this places greater requirements on the hardware. The B spline method [ 250] can also be used to trim a contour.. It requires less high-performance hardware but may lead to drops in path velocity on less properly programmed contours.

If these HSC programs also contain circular blocks, the transitions can be smoothed by \#CONTOUR MODEfunctions. This requires the option CIR_MODE [ 253] and the channel parameter P-CHAN-00239 which are described in the sections mentioned above. HSC programs may also include circular blocks. If NC programs contains many short blocks, it is advisable to use the HSC profile generator \#SLOPE[TYPE=HSC] [> 372].

Besides smoothing a programmed contour, a frequent function is to filter axis command values symmetrically. These functions are described in Section Filter programming.
Besides these recommended standard methods, there are a number of other methods such as interpolation with the Akima spline [ 291], the direct programming of B spline control points [> 297] and older HSC functions [ 300].
\begin{tabular}{|l|l|l|l|}
\hline Name of function & Its suitability & Advantages & Disadvantages \\
\hline \#CONTOUR MODE & \begin{tabular}{l} 
For simple contours \\
with few long blocks
\end{tabular} & \begin{tabular}{l} 
Greater path velocities \\
at contour knee \\
angles
\end{tabular} & Not for short blocks \\
\hline SURFACE methods & \begin{tabular}{l} 
For complex contours \\
with several short \\
blocks
\end{tabular} & Very rugged & \begin{tabular}{l} 
Increased hardware \\
requirements
\end{tabular} \\
\hline B spline method & Trim a contour & \begin{tabular}{l} 
Slow motion sections \\
with unfavourable pro- \\
gramming
\end{tabular} & \begin{tabular}{l} 
Not so high hardware \\
requirements
\end{tabular} \\
\hline Filter programming & \begin{tabular}{l} 
To filter axis com- \\
mand values symmet- \\
rically
\end{tabular} & \begin{tabular}{l} 
Interpolate specified \\
interpolation points
\end{tabular} & \begin{tabular}{l} 
Runs precisely \\
through the pro- \\
grammed points
\end{tabular} \\
\hline Akima spline & \begin{tabular}{l} 
Generally requires a \\
denser and exactly \\
calculated specifica- \\
tion of points
\end{tabular} \\
\hline \begin{tabular}{l} 
PSC functions with \\
OP1 and OP2
\end{tabular} & Rigid machines & \begin{tabular}{l} 
Low hardware require- \\
ments
\end{tabular} & \begin{tabular}{l} 
Relatively strong ex- \\
citation of machine \\
structure
\end{tabular} \\
\hline
\end{tabular}

\subsection*{11.1 Programs with several short blocks}

\section*{Notice}

The use of this feature requires a license for the "HSC" extension package. It is not included in the scope of the standard license.

\section*{Attention}

B splines for the programmed control points are generated using extended HSC programming. For this kind of HSC programming it is recommended to first select the HSC profile type (slope 3) using the command \#SLOPE [TYPE...] [ 372].

Depending on the machining task the following 2 methods are available for selecting/deselecting HSC programming and parametrisation:

Method 1 is especially suited to a single move around a contour (trimming). In this case, the contour consists of very many short blocks which are to be moved at a high feedrate.


Fig. 67: Trim a contour

Method 2 is especially suited to machining free-form surfaces. For NC programs generated by CAD systems, the workpiece is usually machined in several paths (line-by-line or helical). Special algorithms are used (surface optimiser) to achieve a high surface quality within the shortest possible machining time.


Fig. 68: Line-by-line surface machining
\begin{tabular}{|c|c|}
\hline \multicolumn{2}{|r|}{Trim a contour (\#HSC ON/OFF)} \\
\hline \begin{tabular}{l}
\#HSC [ON | OFF] [ [ BS \\
[ME \\
[AU \\
[AU \\
[MA
\end{tabular} & ```
E [PATH_DEV=..] [TRACK_DEV=..]
=..]
OFF_PATH=..] [AUTO_OFF_TRACK=..]
OFF_G00=..] [AUTO_OFF_G60=..]
ATH_LENGTH=..] [MAX_ANGLE=..] ]]
``` \\
\hline ON & Enable HSC programming. \\
\hline OFF & Disable HSC programming. \\
\hline BSPLINE & Keyword for HSC programming with BSPLINE. Must always be programmed as first keyword. \\
\hline PATH_DEV=.. & \begin{tabular}{l}
Maximum deviation of \(B\) spline from programmed path contour in [mm, inch *]. The spline is deselected automatically if this deviation is exceeded. If the maximum deviation is defined as 0 , path deviation is not monitored. \\
Default value: 0.2 mm \\
*when P-CHAN-00439 is active
\end{tabular} \\
\hline \multirow[t]{2}{*}{TRACK_DEV=..} & Maximum deviation of tracking axes in [ \({ }^{\circ}\) ]. If the maximum deviation is defined as 0 , tracking axes is not monitored. \\
\hline & Default value: \(5^{\circ}\) \\
\hline \multirow[t]{3}{*}{MERGE=..} & Merge blocks. The maximum deviation is determined depending on the values taken from PATH_DEV and TRACK_DEV. \\
\hline & 0 : No block merging (default) \\
\hline & 1: Merge blocks \\
\hline \multirow[t]{3}{*}{AUTO_OFF_PATH=..} & Automatic block separation if the programmed \(B\) spline deviation of the main axes is exceeded (PATH_DEV). \\
\hline & 0 : No deselection if deviation is too large (default), block is separated \\
\hline & 1: Deselect if deviation is too large \\
\hline \multirow[t]{3}{*}{AUTO_OFF_TRACK=..} & Automatic block separation if the programmed \(B\) spline deviation of the tracking axes is exceeded (TRACK_DEV). \\
\hline & 0 : No deselection if deviation is too large (default), block is separated \\
\hline & 1: Deselect if deviation is too large \\
\hline \multirow[t]{3}{*}{AUTO_OFF_G00=..} & Automatic deselection of B spline interpolation for G00 blocks. \\
\hline & 0 : No implicit deselection due to rapid traverse block (default) \\
\hline & 1: Implicit deselection due to a rapid traverse block \\
\hline \multirow[t]{3}{*}{AUTO_OFF_G60=..} & Automatic deselection of B Spline interpolation for programmed exact stop G60 or G360. \\
\hline & 0: No implicit deselection due to exact stop (default) \\
\hline & 1: Implicit deselection due to exact stop \\
\hline \multirow[t]{2}{*}{MAX_PATH_LENGTH=..} & Minimum path length of relevant blocks in [mm, inch *]. If blocks are longer than the specified length, the B Spline is deselected implicitly. \\
\hline & Default value: 0 mm (implicit deselection due to block length does not take place) *with active P-CHAN-00439 \\
\hline
\end{tabular}

Maximum contour knee angle in [ \({ }^{\circ}\) ] for transitions between two linear blocks up to which a B spline is inserted. The B Spline is deselected internally if the angle between two linear blocks is greater.
Default value: \(160^{\circ}\)


Control points are programmed with linear blocks (G00 and G01). Their target points are used as control points. It must be considered that only the start and end of the curve runs straight through the control points.

\section*{Notice}

The parameters may also be specified in several steps. For example, this means that it is possible to first define the maximum contour deviation ("PATH_DEV "). Then in a second command, the maximum path length ("MAX_PATH_LENGTH ") and selection of B spline interpolation ("ON") are defined.

\section*{Attention}

Parameterisation cannot be changed while B spline interpolation is active.

\section*{Programing Example}

\section*{Trimming a contour}

The spline curve is based on the control points N40-N155 but in this case, the spline curve only runs straight through them at N20 and N150.

N20 G00 X0 Y0 Z0 F10000
N30 \#HSC ON [BSPLINE PATH_DEV=0.2 MERGE=1 ...] Parametrisation + selection
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12

N70 X25 Y25
N80 X30 Y35
N90 X50 Y37.5
N100 X55 Y32.5
N110 X58 Y12
N120 X70 Y12
N130 X77.5 Y10
N140 X90 Y35
N150 X100 Y37.5
N160 \#HSC OFF
N170 M30
... or also
```

N20 G00 X0 Y0 Z0 F10000
N25 \#HSC [BSPLINE PATH_DEV=0.2 MERGE=1 ...] Parameterisation
N30 \#HSC ON Selection
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 X50 Y37.5
N100 X55 Y32.5
N110 X58 Y12
N120 X70 Y12
N130 X77.5 Y10
N140 X90 Y35
N150 X100 Y37.5
N160 \#HSC OFF Deselect
N170 M30

```

\subsection*{11.1.2 Surface machining with Surface Optimiser (method 3)}
```

\#HSC [ON | OFF][ [ SURFACE [PATH_DEV=..][PATH_DEV_G00=..]
[TRACK_DEV=..] [TRACK_DEV_G00=..]
[MAX_ANGLE=..] [CHECK_JERK=..]
[AUTO_OFF_GOO=..] [CIR_MODE=..]
[CIR_MIN_ANGLE=..] [CIR_MIN_RADIUS=..]
[MERGE=..][LENGTH_LONG_CIR=..]]] modal
ON

```

OFF
SURFACE

PATH_DEV=..

Enable HSC programming.
Disable HSC programming.
Keyword for HSC machining with surface optimiser. Must always be programmed as first keyword.
Define maximum contour error.
\(>0.0\) : Maximum path deviation in [mm, inch *]
Default value: 0.2 mm

\section*{Notice}

Empirically, the contour error should be set 2 or 3 times larger than the secant error which is defined when the NC program is generated in the CAM system.
The tool is not in contact with the workpiece in G0 motions. As a result, the tolerance can be set significantly larger than PATH_DEF without changing the precision of the workpiece.
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{PATH_DEV_G00=..} & Define maximum contour error for G0-G0 transitions. \\
\hline & > 0.0: Maximum path deviation in [mm, inch *] \\
\hline & Default value: The value of PATH_DEV applies \\
\hline \multirow[t]{2}{*}{TRACK_DEV=..} & Define the maximum orientation error. \(>=0.0\) : Maximum path deviation in [ \({ }^{\circ}\) ] \\
\hline & Default value: \(2^{\circ}\) \\
\hline \multirow[t]{2}{*}{TRACK_DEV_G00=..} & Define the maximum orientation error for G0-G0 transitions \(>=0.0\) : Maximum path deviation in [ \({ }^{\circ}\) ] \\
\hline & Default value: The value of TRACK_DEV applies \\
\hline
\end{tabular}

\section*{Notice}

If a ball milling cutter is used, the value can be set significantly larger than PATH_DEV (e.g. 10 times).
The tool is not in contact with the workpiece in G0 motions. As a result, the tolerance can be set significantly larger than TRACK_DEV without influencing the precision of the workpiece.

MAX_ANGLE=..

Define the maximum contour knee angle in [ \({ }^{\circ}\) ] for transitions between two linear blocks up to which the this mode can be used. If the angle between the two linear blocks exceeds this limit, the mode is deselected internally.
\(>=0.0\) : Maximum knee angle in [ \({ }^{\circ}\) ]
Default value: \(160^{\circ}\)


CHECK_JERK=.. Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110). This parameter overwrites the initial state defined in the channel parameter list by P-CHAN-00110 (check_jerk_on_poly_path).

0: No jerk monitoring
1: Jerk monitoring based on the geometric ramp time P-AXIS-00199. This may reduce path velocity.
2: Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non-linear velocity profile.
AUTO_OFF_G00=.. Automatic deselection of optimisation with G00 blocks
0 : No implicit deselection due to rapid traverse block (default)
1: Implicit deselection due to a rapid traverse block
CIR_MODE=..

CIR_MIN_ANGLE=..

CIR_MIN_RADIUS=..

Define contouring of circular motions:
0 : No contouring of circular motions G02/G03
1 : Circular block contouring. (default)
2 : Contour circular blocks and optimise long circular blocks
Available as of V3.1.3075.01
Define minimum circle angle
Valid values: >= 0.0 : Minimum circle angle in degrees
The minimum circle angle at which circular motions can be traversed by the method using exact interpolation. Circular blocks with small swept angles are approximated by a spline curve for faster processing. (Default value \(=30^{\circ}\) )
Define minimum circle radius
Valid values: >= 0.0 : Maximum circle radius in [mm, inch *]
The minimum circle radius defines the circle radius from which circular motions can be traversed by the method using exact interpolation. Circular blocks with a small radius or with the magnitude of PATH_DEV are approximated by a spline curve.
Available as of V3.1.3075.01

MERGE=..
```

from PATH_DEV and TRACK_DEV.
0 : No block merging (default)
1: Merge blocks
LENGTH_LONG_CIR=.. Minimum length of segments for long circular blocks when CIR_MODE= 2 in [mm, inch *] is used
(default value=2)
Available as of V3.1.3075.01
*with active P-CHAN-00439

```

Default values of free-form surface machining
\begin{tabular}{|l|l|}
\hline PATH_DEV & 0.2 mm (default value of PATH_DEV) \\
\hline TRACK_DEV & \(2^{\circ}\) (default value of TRACK_DEV) \\
\hline PATH_DEV_G00 & PATH_DEV \\
\hline TRACK_DEV_G00 & TRACK_DEV \\
\hline CIR_MODE & 1 \\
\hline MAX_ANGLE & \(160^{\circ}\) \\
\hline CHECK_JERK & \begin{tabular}{l} 
The valid channel parameter is P-CHAN-00110 \\
(check_jerk_on_poly_path, default value = 1)
\end{tabular} \\
\hline AUTO_OFF_G00 & 0 \\
\hline CIR_MIN_ANGLE & \(30^{\circ}\) \\
\hline CIR_MIN_RADIUS & 0.0 \\
\hline LENGTH_LONG_CIR & 2 mm \\
\hline
\end{tabular}

\section*{Notice}
i
The parameters may also be specified in several steps. For example, this means it is possible to first define the maximum contour deviation ("PATH_DEV "). Then in a second command, jerk monitoring ("CHECK_JERK") and the selection of HSC surface interpolation ("ON") are defined.

\section*{i \\ Notice \\ When \#HSC[SURFACE] is used, you are advised to use \#SLOPE[TYPE=HSC] at the same time for path velocity planning.}

\footnotetext{
\section*{Attention}

Parameterisation cannot be changed while smoothing is active.
One condition to use this function is that it should be parameterised in the start-up list for each channel in which the function is to be used.
}


\subsection*{11.1.3 FIR filter (\#FILTER)}

\section*{Release Note}

The availability of this function depends on the configuration and on the version scope.

In order to achieve a high surface finish in free-form surface machining, any excitation of machine oscillations must be avoided as far as possible.
FIR axis filters (Finite Impulse Response filters) provide the user with the option of smoothing the axis setpoints for the drives to minimise excitations in the machine.

The precondition for using a FIR filter using the \#FILTER command is a configured filter type (P-AXIS-00586) of the corresponding axis.

\section*{Notice}

This function is an additional option requiring a license.
```

\#FILTER [ON | OFF] [ORDER=.. ORDER_TIME=.. SHARE=.. AX_DEV=..
FCUT=.. ACC_FACT=.. QUALITY=.. ]

```
\begin{tabular}{|c|c|}
\hline ON & Enable FIR filter. \\
\hline OFF & Disable FIR filter. \\
\hline ORDER=<expr> & Specify filter order \\
\hline ORDER_TIME=<expr> & Specify filter order over time [ \(\mu \mathrm{s}\) ] \\
\hline SHARE=<expr> & \begin{tabular}{l}
Define the degree of effectiveness (analogous to P-AXIS-00590) of the filter [\%] value range \(0-100\) \\
default value \(=100\)
\end{tabular} \\
\hline AX_DEV=<expr> & \begin{tabular}{l}
Specify the tolerance for tolerance monitoring in [mm, inch *]. Default value \(=0\) (no tolerance monitoring). \\
*with active P-CHAN-00439
\end{tabular} \\
\hline FCUT=<expr> & Specify the cut-off frequency (analogous to (P-AXIS-00585) of the filter [Hz] default value \(=30\) \\
\hline ACC_FACT=<expr> & Increase the path velocity at block transitions with FIR filter enabled. \\
\hline & The greater the value setting, the less the velocity is reduced at the block transition. \\
\hline & This requires a valid setting of P-AXIS-00013 (a_trans_weight) for the axes. \\
\hline & Value range \(=1.0-10.0\) \\
\hline & Default value =: 1.0 \\
\hline QUALITY=<expr> & \begin{tabular}{l}
Specify the filter quality of the filter core curve value range: \(0<\) QUALITY <= 1 \\
default value \(=1.0\)
\end{tabular} \\
\hline & Parameter available as of V3.1.3075.04 \\
\hline
\end{tabular}

\section*{Notice}

The \#FILTER ON/OFF command enables or disables all the FIR filters of the axes in the channel.

It is possible to use FIR filters on all axes. It is also possible to use different filters for each axis by axis-specific configuration in the axis lists.
FIR filters can be globally enabled or disabled and reparameterised across all axes in the NC program during machining (see Programming example).

\section*{Notice}

Tolerance monitoring can only be configured and activated in the NC program.

Tolerance monitoring is programmed by the parameter AX_DEV. It ensures that every axis remains within the specified tolerance [mm, inch].
Tolerance monitoring always monitors all axes and therefore can only be controlled globally in the NC program.
Tolerance monitoring in only active if AX_DEF was specified with a corresponding tolerance. For further information see [FCT-C37//Description]

This command replaces the previously available \#FILTER ON [HSC] command.

\subsection*{11.2 Polynomial contouring for long blocks (G61/G261/G260)}
\begin{tabular}{|lcc|}
\hline G61 & Polynomial contouring (at block end) & (non-modal) \\
\(\ldots\) or for polynomial contouring across several blocks: & \\
G261 & Select polynomial contouring (at block end) & (modal) \\
G260 & Deselect polynomial contouring & (modal) \\
\hline
\end{tabular}

\subsection*{11.2.1 Definition of terms}

The following terms are briefly explained:

Polynomial contouring:
Contouring curve:
Block length:
Corner distance:

Curvature and direction-continuous connection of two motion blocks.
Curve composed of two 4th order polynomials per axis.
The path length of the curve corresponding to the motion block.
Distance from the start/end of the contouring curve to the programmed target point/ starting point of a motion block (see figure below). The corner distance is always limited to half the block length. In a circular block, the corner distance is the arc length from the starting point of the contouring curve up to the programmed target point of the arc.


Fig. 69: Definition of corner distance

Pre-block: Motion block before the contouring curve
Post-block:
Pre-distance:
Post-distance:
Interim point:
Corner deviation:
Motion block after the contouring curve
Corner distance of the pre-block
Corner distance of the post-block
Point at which the two partial curves of the contouring curve meet.
The distance between the programmed corner point and the interim point of the contouring curve (see figure below).


Fig. 70: Definition of corner deviation

\section*{Programing Example}

\section*{Comparing the programming of G61 - G261/G260}

The 3 NC programs all generate the identical contour shown in the figure below.
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline \multicolumn{3}{|l|}{\%poly_G61} & \multicolumn{3}{|l|}{\%poly_G261_1} & \multicolumn{3}{|l|}{\%poly_G261_2} \\
\hline N10 & X 0 & Y0 G01 F1000 & N10 & X 0 & Y0 G01 F1000 & N10 & X0 & Y0 G01 F1000 \\
\hline N20 & X20 & Y100 & N20 & X20 & Y100 & N20 & X20 & Y100 \\
\hline N30 & G61 & X40 Y100 & N30 & G261 & X40 Y100 & N25 & G261 & \\
\hline N40 & G61 & X60 Y20 & N40 & X60 & Y20 & N30 & X40 & Y100 \\
\hline N50 & G61 & X80 Y20 & N50 & X80 & Y20 & N40 & X60 & Y20 \\
\hline N60 & G61 & X100 Y100 & N60 & X100 & Y100 & N50 & X80 & Y20 \\
\hline N70 & X120 & Y100 & N70 & G260 & X120 Y100 & N60 & X100 & Y100 \\
\hline N80 & X140 & Y20 & N80 & X140 & Y20 & N70 & X120 & Y100 \\
\hline N90 & X160 & Y20 & N90 & X160 & Y20 & N75 & G260 & \\
\hline N100 & M30 & & N100 & M30 & & N80 & X140 & Y20 \\
\hline
\end{tabular}


Fig. 71: Contour from programming G61 - G261/G260
kerne

\subsection*{11.2.2 General properties}

The process of polynomial contouring is calculated from the geometrical path contour of the main axes in space. The given conditions, e.g. corner deviation or percentage path velocity, result in a position on the original contour from which the contour can be changed or replaced by a contouring curve (polynomial). This means that the starting or target point of the contouring curve which is known on the original path curve.
Using the determined starting and target points of the polynomial of the main axes calculated using the conditions, it is also possible to specify the position of the tracking axes at which their original contour can be replaced by a polynomial.
With tracking axes, as with main axes, a curvature and direction-continuous polynomial is inserted between the corner distances of the pre-block and post-block, taking into consideration the max. acceleration of these axes. However, the originally specified corner deviation refers only to the deviation of the main axis in space so that, if required, an additional limit value can be specified for the maximum deviation of the tracking axes. Any theoretical excess of this deviation by the tracking axis causes a reduction in the contouring curve (reduction in corner distance).

Polynomial contouring is automatically suppressed depending on the transition between the preblock and post-block if:
- The transition of all axes is mirrored tangentially or directly.
- The transition of the main axes is tangential and no maximum deviation (value \(=0\) ) was specified for the tracking axes.
- After programming G61, program end is reached without post-block. In addition, a warning message is output.

\subsection*{11.2.2.1 Maximum corner distance, minimum residual block length}

To avoid any "degeneration" of the polynomial, the following limitations apply additionally:
- The corner distance may assume a maximum of \(50 \%\) of the original block length. If the corner distance selected is greater, the distance of the pre-block and post-block is limited accordingly. If the corner distance at block start and block end is \(50 \%\) of the original block length, the block is skipped completely.
- When parameterising the contouring function, the minimum residual block length can be set between \(0 \%\) and \(100 \%\). This corresponds to a variable maximum corner distance from \(50 \%\) to \(0 \%\). At every program start, the minimal residual block length is first set to 0\% (block can be completely contoured). If the minimum residual block length is specified as \(10 \%\), for example, the corner distances of this block can be maximum ( \(100 \%-10 \%\) ) / \(2=45 \%\) of the original block length.
- In circular blocks the maximum corner distance (distance travelled on circle) is limited so that the angle covered does not exceed \(90^{\circ}\).

\subsection*{11.2.2.2 Relevant block length}

As a rule, the minimum curve length is limited because of the mathematical resolution of the controller (given by the number of significant digits of a REAL value). At the moment, this length is 31.7 micrometres. Path segments less than this minimum curve length generate an error message and result in abortion of the contouring function at the current block transition.

In addition, the contour can include very short compensation blocks which are inserted by a programming system (CAD/CAM) or by tool radius compensation. After compensation the block retains a continuous path.

To avoid abortion of contouring by these short blocks, a minimum block length can be defined. From this point onwards, polynomial contouring is then relevant for the post-block. Shorter blocks are skipped during active contouring, i.e. contouring is considered in the following block.

Here, a limit for the motion path of the main axes in space as well as a limit can be specified for the motion path of the tracking axes. The block is skipped completely only when both the motion path of the main axes and the motion path of the individual tracking axes are below the specified limit. Polynomial contouring combines the pre-block and the post-block in direction and curvaturecontinuous function. The initial blocks need not be adjacent (contour need not be continuous).

If a block is skipped, the maximum corner deviation of the main axes and tracking axes can only be approximated. This means, it is assumed that the deviation of the contouring can be ignored in the skipped blocks.

\section*{Programing Example}

\section*{Relevant block length}
```

\#CONTOUR MODE [DEV, PATH_DEV 5, RELEVANT_PATH 2]
N03 G01 X0 Y0 Z0 C0 F4
N907090 G04 X0.1
N04 X5 G261
N05 Y1
N09 X10 Y3 G260
N907091 Y0

```


Fig. 72: Example of skipping a short block N05 when contouring

Special case 1: Sequence of multiple short blocks behind the block transition
If multiple sequential blocks (N20, N30, N40) are shorter than the minimum motion path specified, the blocks are skipped provided the distance to the target point from the last relevant end point ( N 10 ) is shorter than the specified minimum motion path. If the target point of the skipped block is outside this envelope curve, the block (N40) is used to calculate the contouring curve even if it is shorter than the specified minimum length. This method permits a slight deviation from the original contour even if multiple sequential blocks are skipped.


Fig. 73: Some single blocks ( \(\mathrm{N} 2 \mathrm{O}, \mathrm{N} 30\) and N 40 ) are too short but the target point is outside the minimum block length.

Special case 2: Sequence of multiple short blocks behind the block transition, last block is extremely short

As an exception, the block N40 itself may be shorter than the minimum system-specific length (about 15 micrometres) required for contouring. In this case, the last end point and the new target point are connected by a linear block. This new linear block \(N 20^{\prime}\) is then used to calculate the contouring curve.

\section*{Target}


Fig. 74: Single blocks (N20, N30 and N40) are too short but the sum of all blocks exceeds the minimum system-specific block length.

Special case 3: Short blocks before the block transition
If the blocks at the beginning of contouring (before block transition) are already shorter than the minimum system-specific length, the blocks are skipped. The blocks are skipped until the distance between the last valid point and the current target point exceeds the minimum block length. If this is the case, the last end point and the current target point is connected by a linear block N10'. This linear block then is used as the start block for contouring.

\section*{Target}


Fig. 75: Multiple blocks (N10, N20 and N30) are too short but the sum of all blocks exceeds the minimum system-specific block length.

Special case 4: Deselecting contouring or changing parameterisation
If contouring is deselected when blocks are skipped or the basic conditions for contouring are changed, the current contouring may only be continued up until they are deselected or parameters are changed. After this function, contouring may be continued using the new parameters.
\#CONTOUR MODE [ DEV, PATH_DEV 5, RELEVANT_PATH 2] N10 G91 G01 F1000 X10 G61
N20 X2 Y1
N30 Y1. 5
N40 X-1 Y2...


Fig. 76: Some single blocks ( \(\mathrm{N} 20, \mathrm{~N} 30\) and \(\mathbf{N} 40\) ) are too short but contouring is deselected as of block N20.

\subsection*{11.2.2.3 Executing additional blocks}

If a command without contour information is programmed in addition to the motion blocks at block end (N10 - N20) (e.g. M function requiring acknowledgement with pre-block output and post-block synchronisation, MVS_SNS), the command may be executed before, during or after the contouring curve.

\section*{Programing Example}

\section*{Executing additional blocks}

N10 X100 G61 M25
N20 Y100


Fig. 77: Synchronisation without contour-relevant actions during contouring


Fig. 78: Synchronisation without contour-relevant actions after contouring

There are 3 options to execute these commands:
1. Directly after pre-block (N10) and before the first contouring polynomial
2. Between the first and the second contouring polynomial
3. After the second contouring polynomial and before the post-block (N20)

\subsection*{11.2.2.4 Jerk within the polynomial}

The curvature of the polynomial results in a jerk for the axes running across the path trajectory. This jerk is normally checked with the maximum dynamic parameters of the axes (P-AXIS-00199). If the jerk is too strong, path velocity is reduced accordingly. In some user-specific applications, this reduction in velocity is undesirable because of the maximum jerk. This can be defined specifically by control commands in the NC command \#CONTOUR MODE. The control commands overwrite the pre-definition in the channel parameter list P-CHAN-00110 and are valid modal up to program end.
In the example below, the block transition from N6 to N7 is contoured by polynomials and this is considered by the jerk. The transition from N7 to N8 is also contoured but with no consideration for jerk on the path contour.

\section*{Programing Example}
\%poly_jerk.nc
(default setting in the channel parameter list: (check_jerk_on_poly_path)
\#SLOPE [TYPE=TRAPEZ]
\#CONTOUR MODE [ DEV, PATH_DEV 4, RELEVANT_PATH 51]
N0003 G1 X0 Y100 Z0 F4

N0004 G261
N0005 G1 G91 X100
N0006 Y-50
NOOO7 \#CONTOUR MODE [CHECK_JERK=1]
N0008 X100
NOOO9 \#CONTOUR MODE [CHECK JERK=0]
N0010 Y-50
N0009 G260
N0055 M30

\subsection*{11.2.2.5 Velocity curve in the contouring section}

Depending on axis parameterisation and the application, it may be necessary to influence the velocity curve in the contouring section. In the default definition, the contouring section is travelled at maximum permissible path velocity. If the axes have strongly different dynamics, this could lead to an unacceptable excitation of vibrations in the machine because path velocity is adjusted in the contouring section.

The characteristic in the contouring section can be adjusted by specific control commands in the NC command \#CONTOUR MODE.

In the example below, the block transition from N6 to N7 is contoured by polynomials which are moved in the contouring section at maximum velocity, i.e. the velocity is adjusted here by different axis dynamics. The transition from N9 to N10 is also contoured but without any velocity adjustment. This leads to a constant path velocity in the contouring section.

\section*{Programing Example}
```

%poly_const_speed
NOOO3- \#SLOPE [TYPE=TRAPEZ]
N0004 G1 X0 Y0 Z0 F8000
NOOO5 \#CONTOUR MODE [CONST_VEL=0]
N0006 X100 G61
N0007 Y100
NOOO8 \#CONTOUR MODE [CONST_VEL=1]
N0009 X0 G61
N0010 Y0
N0020 M30

```


Fig. 79: Characteristic in the transition section

\subsection*{11.2.3 Parameterising contouring modes in the NC program (\#CONTOUR MODE)}

Before the actual activation of polynomial contouring (G61/G261), the individual options are parameterised by the NC command \#CONTOUR MODE.
Depending on the contour mode, specific keywords are provided for parameterisation. The command has the following syntax structure:
\#CONTOUR MODE [<contour_mode> <parameter> <action> ]
\begin{tabular}{|l|l|l|}
\hline <contour_mode> & DEV & Contour corner deviation (default) \\
\cline { 2 - 6 } & DIST & Contour corner distance \\
\cline { 2 - 3 } & DIST_SOFT & Dynamic optimised contouring \\
\cline { 2 - 3 } & DIST_MASTER & Dynamic optimised contouring with master axis \\
\cline { 2 - 3 } & POS & Contour with interim point \\
\cline { 2 - 3 } & PTP & Dynamically optimised contouring of the contour. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline <Parameter> & \begin{tabular}{l} 
PATH_DEV \\
TRACK_DEV \\
\(\ldots\)
\end{tabular} & \begin{tabular}{l} 
Caution: \\
The parameters for deviations and tolerances must al- \\
ways be specified in [mm, inch] or [ \({ }^{\circ}\) ]. When specifying in \\
[inch], please refer to the note in P-CHAN-00439.
\end{tabular} \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline <action> & PRE_ACTION & \multirow{3}{*}{ Execute \(M / H\) actions related to the contouring curve } \\
\cline { 2 - 2 } & INTER_ACTION & \\
\cline { 2 - 2 } & POST_ACTION & \\
\hline
\end{tabular}

\subsection*{11.2.4 Activating contouring modes in the NC program}

Contouring is activated by the G functions G61 (blockwise) or G261 (modal) after parameterising the corresponding contouring mode.

\section*{Release Note}

Starting at Build V2.11.2022.13 and higher
... alternatively, contouring may be selected or deselected by additionally specifying ON/OFF in the command \#CONTOUR MODE. Programming G261/G260 is then no longer necessary.

\section*{Programing Example}

\section*{Activating contouring modes in the NC program}
```

%Contour_on_off
N10 G90 G01 X0 Y0 Z0 A0 C0 F60
N20 \#CONTOUR MODE ON [DEV PATH_DEV=1.0] ; Parameterisation and
;activation (= G261)
N30 X100
N40 Y100
N50 X0
N60 Y0
N70 \#CONTOUR MODE OFF ;Deactivation (= G260)
N80 M30

```

\subsection*{11.2.4.1 Contouring with corner deviation}

\section*{Notice}

Default parameterisation of this contouring type becomes effective after program start.

Corner distances used to shorten motion blocks are automatically determined after purely geometric considerations to prevent a user-specified corner deviation from being exceeded.

Corner distances are limited depending on the specified minimum residual block length. However, both distances are limited symmetrically. In this case, the programmed path velocity has no influence on the contouring curve.

It is better to use the parameter RELEVANT_PATH to obtain optimised contouring. It is recommended to adopt the maximum corner deviation value PATH_DEV.
Parameterisation takes place as follows:
```

\#CONTOUR MODE [ DEV [PATH_DEV<expr>] [RELEVANT_PATH<expr>]
[RELEVANT_TRACK<expr>] [TRACK_DEV<expr>]
[REMAIN_PART<expr>] [<action>]
[CHECK_JERK<expr>] [MAX_ANGLE<expr>]
[CONST_VEL<expr>]]

```
\begin{tabular}{ll} 
DEV & Contour with maximum corner deviation \\
PATH_DEV<expr> & \begin{tabular}{l} 
Maximum deviation of programmed contour in [mm, inch \(]^{*}\) \\
Default value: 1 mm
\end{tabular} \\
RELEVANT_PATH<expr> & \begin{tabular}{l} 
Minimum path length of relevant post-blocks in [mm, inch *]. \\
Default value: 0 mm
\end{tabular} \\
RELEVANT_TRACK<expr> & \begin{tabular}{l} 
Minimum path of tracking axis for relevant post-blocks in [ \({ }^{\circ}\) ] \\
\\
Default value: \(0^{\circ}\)
\end{tabular} \\
TRACK_DEV<expr & \begin{tabular}{l} 
Maximum deviation of tracking axes in [ \(\left.{ }^{\circ}\right]\) \\
REMAIN_PART<expr>
\end{tabular} \\
\begin{tabular}{l} 
Default value: \(0^{\circ}\) \\
Distance to go in [0\%-100\%] of original block \\
* when P-CHAN-00439 is active
\end{tabular} & Default value: \(0 \%\)
\end{tabular}
\begin{tabular}{|l|l|}
\hline Keywords & Meaning \\
\hline PRE_ACTION & Actions before contouring curve. \\
\hline INTER_ACTION & Actions in contouring curve (default). \\
\hline POST_ACTION & Actions after contouring curve. \\
\hline
\end{tabular}

CHECK_JERK<expr>
Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110).
\begin{tabular}{|l|l|}
\hline Value & Meaning \\
\hline 0 & Without jerk monitoring (default). \\
\hline 1 & \begin{tabular}{l} 
Jerk monitoring based on the geometric ramp time (P-AXIS-00199). This may reduce \\
path velocity.
\end{tabular} \\
\hline 2 & \begin{tabular}{l} 
Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non- \\
linear velocity profile.
\end{tabular} \\
\hline
\end{tabular}

MAX_ANGLE<expr>
Maximum contour knee angle in [ \({ }^{\circ}\) ] for transitions between two linear blocks up to which contouring is active. If a circular block is included at the transition, the bend angle is not evaluated.
Default value: \(178^{\circ}\) (i.e. the entire contour is contoured)


CONST_VEL<expr> Constant path velocity in the contouring section.

\section*{Value Meaning}
\begin{tabular}{|l|l|}
\hline 0 & Without constant path velocity (default). \\
\hline 1 & At constant path velocity. \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Contour with corner deviation}

N100 \#CONTOUR MODE [DEV PATH_DEV=5]
N110 G01 X100 G61
N120 G01 Y100


Fig. 80: Contouring with corner deviation

\subsection*{11.2.4.2 Corner distance contouring}

If the point from which the original contour may be left is known, the user can explicitly specify the corner distances of the pre-blocks and post-blocks by which the adjacent motion blocks are to be shortened.

The corner distances are limited if they do not exceed the minimum residual block length .
If the corner distances a and b are specified as identical, the other corner distance is symmetrically limited if one corner distance is limited to the minimum remaining block length.

If the corner distances \(a\) and \(b\) are specified as different, only the distance which is too long is reduced if limited. With asymmetrical path lengths, this can lead to a "degenerated" contour but this may sometimes be desirable.

Parameterisation takes place as follows:
```

\#CONTOUR MODE [ DIST [PRE_DIST<expr>] [POST_DIST<expr>]
[RELEVANT_PATH<expr>] [RELEVANT_TRACK<expr>]
[TRACK_DEV<expr>] [REMAIN_PART<expr>] [<action>]
[CHECK_JERK<expr>][MAX_ANGLE<expr>]
[CONST_VEL<expr>]]

```

DIST
PRE_DIST<expr>

POST_DIST<expr>

RELEVANT_PATH<expr>

RELEVANT_TRACK<expr>

TRACK_DEV<expr>

REMAIN_PART<expr>
*when P-CHAN-00439 is active
<action>

Contour with corner distance specified
Corner distance in [mm, inch* after which there is a deviation from the original contour. Default value: 1 mm
Corner distance in [mm, inch* after which there is a return to the original contour. Default value: 1 mm

Minimum path length of relevant post-blocks in [mm, inch *]
Default value: 0 mm
Minimum path length of tracking axis for relevant post-blocks in [ \({ }^{\circ}\) ].
Default value: \(0^{\circ}\)
Maximum deviation of tracking axes in [ \({ }^{\circ}\) ]
Default value: \(0^{\circ}\)
Distance to go in [0\%-100\%] of original block
Default value: 0 \%

Time of execution of additional actions (M/H):
\begin{tabular}{|l|l|}
\hline Keywords & Meaning \\
\hline PRE_ACTION & Actions before contouring curve. \\
\hline INTER_ACTION & Actions in contouring curve (default). \\
\hline POST_ACTION & Actions after contouring curve. \\
\hline
\end{tabular}

CHECK_JERK<expr> Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110).
\begin{tabular}{|l|l|}
\hline Value & Meaning \\
\hline 0 & Without jerk monitoring (default). \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline 1 & \begin{tabular}{l} 
Jerk monitoring based on the geometric ramp time (P-AXIS-00199). This may reduce \\
path velocity.
\end{tabular} \\
\hline 2 & \begin{tabular}{l} 
Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non- \\
linear velocity profile
\end{tabular} \\
\hline
\end{tabular}

MAX_ANGLE<expr>
Maximum contour knee angle in [ \({ }^{\circ}\) ] for transitions between two linear blocks up to which contouring is active. If a circular block is included at the transition, the bend angle is not evaluated.
Default value: \(178^{\circ}\) (i.e. the entire contour is contoured)


CONST_VEL<expr> Constant path velocity in the contouring section.
\begin{tabular}{|l|l|}
\hline Value & Meaning \\
\hline 0 & Without constant path velocity (default). \\
\hline 1 & At constant path velocity. \\
\hline
\end{tabular}

\section*{Programing Example}

Corner distance contouring
```

N100 \#CONTOUR MODE [DIST PRE_DIST=10 POST_DIST=5]
N110 G01 X100 G61
N120 G01 Y100

```


Fig. 81: Corner distance contouring
kerne

\subsection*{11.2.4.3 Dynamic optimised contouring}

Contouring types with corner deviation and interim point define the contouring curve by a direc-tion- and curvature-continuous connection between two motion blocks. This contouring curve referred to the axes may result in a fluctuation in acceleration.

When the possible dynamic data is considered with reference to the axes (acceleration, jerk), the contouring curve is defined at uniform acceleration (minimum jerk) of the two axes. By utilising maximum axis acceleration, the duration of the contouring curve is reduced.

Parameterisation takes place as follows:
```

\#CONTOUR MODE [ DIST_SOFT [PATH_DIST<expr>] [TRACK_DIST<expr>]
[ACC_MAX<expr>] [ACC_MIN<expr>] [RAMP_TIME<expr>]
[DIST_WEIGHT<expr>]]

```
\begin{tabular}{|c|c|}
\hline DIST_SOFT & Dynamic optimised contouring \\
\hline \multirow[t]{3}{*}{PATH_DIST<expr>} & Corner distance of pre-block and post-block (symmetrical) in [mm, inch *] after which a deviation from the original contour is allowed. The definition refers to the motion path of the feed axes. \\
\hline & Default value: 1 mm \\
\hline & Monitoring off: -1 mm \\
\hline \multirow[t]{3}{*}{TRACK_DIST<expr>} & Corner distance to pre-block and post-block in [ \({ }^{\circ}\) ] after which non-feed axes (tracking axes) may deviate from the original contour. \\
\hline & Default value: Value is adopted automatically from PATH_DIST provided this value was not explicitly specified (since program start). \\
\hline & Monitoring off: \(-1^{\circ}\) \\
\hline \multirow[t]{2}{*}{ACC_MAX<expr>} & Percentage in [0\%-100\%] of maximum axis acceleration (machine data) which may be used by the contouring curve. \\
\hline & Default value: \(100 \%\) \\
\hline \multirow[t]{2}{*}{ACC_MIN<expr>} & Percentage in [0\%-100\%] of maximum axis acceleration (machine data) which should be used by the contouring curve. If the specified corner distance (see PATH_DIST) is not maintained, the acceleration is increased to maximum value (ACC_MAX). \\
\hline & Default value: 50 \% \\
\hline \multirow[t]{2}{*}{RAMP_TIME<expr>} & Percentage weighting of the ramp time in [0\%-10000\%]. \\
\hline & Default value: 100 \% \\
\hline \multirow[t]{2}{*}{DIST_WEIGHT<expr} & Influences the split of contoured linear blocks in [0\%-100\%]: In the 0\% pre-definition, all blocks are halved; at 100\%, the split ratio corresponds to the lengths of adjacent blocks. This value can be used to combine the two methods by percentage. \\
\hline & Default value: 0 \% \\
\hline
\end{tabular}
*when P-CHAN-00439 is active

\section*{Restrictions:}
- If a circular block is used for contouring, the contouring curve is calculated with corner distance without dynamic optimisation.
- The calculation uses only one ramp time (maximum value of the four individual ramp times).
- No processing of kinematic transformations. In this case, calculation is performed with corner distance without dynamic optimisation.
- In many cases, weighting the corner distances by the parameter DIST_WEIGHT depending on the pre-/post-blocks results in an optimised utilisation of the available block length.

When axis-specific contouring is executed, the corner distances of the pre-block and post-block are always identical (symmetrical). If the maximum corner distances are also limited on the half block motion path, a shorter contouring section and therefore a lower contouring velocity results for longer motion paths due to the shorter preceding/following motion path.


Fig. 82: Maximum corner distance of block N20 independent of the block lengths of N10 and N20 (DIST_WEIGHT = 0\%)

If the length of the pre- and post-blocks are considered in the calculation of the maximum corner distances, the contouring zone can then be increased.


Fig. 83: Maximum corner distance of block N20 subdivided relative to the block lengths of N10 and N30 (DIST_WEIGHT = 100\%)

Programing Example

\section*{Dynamic optimised contouring}

Comparison of contouring of a \(90^{\circ}\) corner with the methods:
- Dynamically optimised contouring (DIST_SOFT):

N010 \#CONTOUR MODE [DIST_SOFT PATH_DIST=12]
N020 G0 X0 Y80
N030 G261
N040 G01 X40 Y40 F2.5
N050 G01 X80 Y80
N060 G260
N070 M30
- Contour with corner deviation (DEV):

N010 \#CONTOUR MODE [DEV PATH_DEV=0.2]
N020 G0 X0 Y80
N030 G261
N040 G01 X40 Y40 F2. 5
N050 G01 X80 Y80
N060 G260
N070 M30
Comparison of contouring curves:


\subsection*{11.2.4.4 Dynamic optimised contouring with master axis}

A feed master axis is used in this variant of the dynamically optimised contouring curve. This generally results in a more favourable velocity profile.
The feed master axis is identified in the axis parameter list by an entry in P-AXIS-00015 and marked as the only feed axis in the channel parameter list (P-CHAN-00011).

Further properties and limitations correspond to the dynamically optimised contouring mode.

Parameterisation takes place as follows:
```

\#CONTOUR MODE [ DIST_MASTER [SYM_DIST<expr>] [ACC_MAX<expr>]
[ACC_MIN<expr>] [RAMP_TIME<expr>]
[DIST_WEIGHT<expr>] ]

```
\begin{tabular}{ll} 
DIST_MASTER & Dynamically optimised contouring with feed master axis \\
SYM_DIST<expr> & \begin{tabular}{l} 
Corner distance of pre-block and post-block (symmetrical) in [mm, inch *] after which a \\
deviation from the original contour is allowed. \\
Default value: 1 mm \\
Monitoring off: -1 mm
\end{tabular} \\
ACC_MAX<expr> & \begin{tabular}{l} 
Percentage in [0\%-100\%] of maximum axis acceleration (machine data) which may be \\
used by the contouring curve.
\end{tabular} \\
ACC_MIN<expr> & \begin{tabular}{l} 
Default value: \(100 \%\) \\
Percentage in [0\%-100\%] of maximum axis acceleration (machine data) which should \\
be used by the contouring curve. If this specified corner distance (see SYM_DIST) is \\
not maintained here, the acceleration is increased up to maximum value (ACC_MAX).
\end{tabular} \\
RAMP_TIME<expr> & \begin{tabular}{l} 
Default value: \(50 \%\)
\end{tabular} \\
Percentage weighting of the ramp time in [0\%-10000\%]. \\
DIST_WEIGHT<expr> & \begin{tabular}{l} 
Default value: \(100 \%\) \\
Percentage weighting of corner distances relative to the pre-/post-block in [0\%-100\%]. \\
\end{tabular} \\
& Default value: \(0 \%\)
\end{tabular}
*when P-CHAN-00439 is active

\subsection*{11.2.4.5 Contour with interim point}

Here, the user specifies not only the corner distances but also an interim point \(P\) at which the two polynomial curves are adjacent to each other (expert mode). This mode permits the retention of the programmed contour and fully utilises the dynamics by specifying the corner distance zero. In other words, the corner distances need not be symmetrical here.

Parameterisation takes place as follows:
```

\#CONTOUR MODE [ POS [PRE_DIST<expr>] [POST_DIST<expr>]
[\mathbf{X<expr>] [Y<expr>] [Z<expr>] [<action>]}
[CHECK_JERK<expr>] [CONST_VEL<expr>]]

```
\begin{tabular}{|c|c|}
\hline POS & Contour by specifying the interim point \\
\hline \multirow[t]{3}{*}{PRE_DIST<expr>} & Corner distance in [mm, inch* after which there is a deviation from the original contour. Return point on the original contour. \\
\hline & Default value: 1 mm \\
\hline & *when P-CHAN-00439 is active \\
\hline \multirow[t]{3}{*}{POST_DIST<expr>} & Corner distance in [mm, inch* after which there is a return to the original contour. Return point on the original contour. \\
\hline & Default value: 1 mm \\
\hline & *when P-CHAN-00439 is active \\
\hline X<expr> & Position of interim point in the first main axis in [mm, inch] \\
\hline Y <expr> & Position of interim point in the second main axis in [mm, inch] \\
\hline Z<expr> & Position of interim point in the third main axis in [mm, inch] \\
\hline <action> & Time of execution of additional actions (M/H): \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline Keywords & Meaning \\
\hline PRE_ACTION & Actions before contouring curve. \\
\hline INTER_ACTION & Actions during the contouring curve (default). \\
\hline POST_ACTION & Actions after contouring curve. \\
\hline
\end{tabular}

CHECK_JERK<expr> Jerk monitoring caused by curvature of the polynomial (cf. P-CHAN-00110).
\begin{tabular}{|l|l|}
\hline Value & Meaning \\
\hline 0 & Without jerk monitoring (default). \\
\hline 1 & \begin{tabular}{l} 
Jerk monitoring based on the geometric ramp time (P-AXIS-00199). This may reduce \\
path velocity.
\end{tabular} \\
\hline 2 & \begin{tabular}{l} 
Jerk monitoring based on ramp times P-AXIS-00195 up to P-AXIS-00198 of the non- \\
linear velocity profile
\end{tabular} \\
\hline
\end{tabular}

\section*{CONST_VEL<expr>}

Constant path velocity in the contouring section.
\begin{tabular}{|l|l|}
\hline Value & Meaning \\
\hline 0 & Without constant path velocity (default). \\
\hline 1 & At constant path velocity. \\
\hline
\end{tabular}

\section*{Programing Example}


Fig. 84: Contour with interim point

\subsection*{11.2.4.6 Dynamically optimised contouring of the complete contour}

This mode is suitable for handling tasks where the feedrate need not be constant in the rounded contour. The contouring curve is selected so that at least one axis involved utilises the dynamics available. As opposed to dynamically optimised contouring (DIST_SOFT), this mode involves the entire contour. The figure below shows a typical application:


Comprehensive planning avoids unnecessary acceleration zeroes at block limits and calculates uniform velocity profiles as shown in the figure below.


Accelerations with constant jerk change to reduce further excitations. The acceleration phases are then placed in the straight sections before and after the rounded contour:


The corner deviation defines the distance of the rounded contour to the program corner point.
If the position is known at which a deviation from the original contour is permitted, the user can explicitly specify the amount of pre-block and post-block corner distances by which the adjacent motion blocks are shortened. The corner distances are limited if they do not exceed the minimum residual block length .

Parameterisation takes place as follows:
```

\#CONTOUR MODE [PTP [PATH_DEV=..] [PATH_DIST=..] [MERGE=..] [<action>] ]

```
PTP Axis-specific contouring with specification of corner distance [as of Build V3.1.3052.01]

PATH_DEV=<expr> Maximum corner deviation from the programmed contour in [ mm , inch *]. Default value: 1 mm
PATH_DIST=<expr> Corner distance of pre-block and post-block (symmetrical) in [mm, inch *] after which a deviation from the original contour is allowed. The definition refers to the motion path of the feed axes [as of Build V3.1.3079.16].
Default value: 1 mm
*when P-CHAN-00439 is active
MERGE=<expr> Merge tangential blocks [as of V3.1.3079.16].
0 : Do not merge
1: Merge (default)
<action> Time of execution of additional actions (M/H):
\begin{tabular}{|l|l|}
\hline Keywords & Meaning \\
\hline PRE_ACTION & Actions before contouring curve. \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline INTER_ACTION & Actions in contouring curve (default). \\
\hline POST_ACTION & Actions after contouring curve. \\
\hline
\end{tabular}

\section*{Attention}

This mode is not suitable for:
a) Programs containing many short motion paths (see also HSC).
b) Programs with circular blocks since this results in the automatic deselection of the mode.

\section*{Attention}

This functionality can only be used if the start-up parameter is parameterised for each channel in which the function is to be used.
Example of a setting in the start-up list:
configuration.channel[].path_preparation.function FCT_DEFAULT|FCT_PTP

Programing Example
Dynamically optimised contouring of the complete contour
```

N100 \#CONTOUR MODE [PTP PATH DEV=5]
N110 G01 X100 G61
N120 G01 Y100

```


Fig. 85: Dyn. optimised contouring of the entire contour specifying corner deviation

\subsection*{11.2.5 Example}

\section*{Programing Example}

The examples below show the influence of the different output of M functions during contouring.
```

N907090 X0 Y0
G91 G01 F6000
NO1 \#CONTOUR MODE [DEV PATH_DEV=10 POST_ACTION]
N10 X100 G61 M25 (MVS_SNS)
N20 Y100 F3000
N30 X100 G61 F6000
N40 G04 X2
N50 Y100
NOO X0 Y0
N60 X100 G61
N70 Y100 M26 (MVS SVS)
N907091 G04 X1

```

Output before the contouring curve:
N01 \#CONTOUR MODE [DEV PATH_DEV=10.0 PRE_ACTION]
\(\times 10^{6}\)
Block limit before contour curve


Fig. 86: Block limit before contouring curve

\section*{Output within contouring curve:}


Fig. 87: Block limit within contouring curve

\section*{Output after contouring curve:}

N01 \#CONTOUR MODE [DEV PATH_DEV=10.0 POST_ACTION]


Fig. 88: Block limit after contouring curve

If the acknowledgement of M25 is delayed by block N10, the motion is stopped after the contouring curve and the program waits for PLC acknowledgement.


Fig. 89: Block limit after contouring curve

\section*{Programing Example}

\section*{Change the limit angle during contouring:}
```

\#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELEVANT_TRACK=0.2]
F10000

```
G261
N5 \#CONTOUR MODE [MAX_ANGLE=3]
N10 G01 X0 Y0 Z0 G61
N15 \#CONTOUR MODE [MAX ANGLE=4]
N20 G01 X100 Y0 Z0
N25 \#CONTOUR MODE [MAX_ANGLE=5]
N30 G01 X100 Y100 Z0
N35 \#CONTOUR MODE [MAX_ANGLE=6]
N40 G01 X0 Y0 Z0 G61
G2 60

\section*{Result:}

Contour the N block always takes place at the limit angle of the previous \(\mathrm{N}<\mathrm{i}-5>\) block.

\section*{Programing Example}

\section*{Variation of the contour angle with constant limit angle:}
```

\#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELEVANT_TRACK=0.2]
\#CONTOUR MODE [RELEVANT_TRACK=0.3]
P100=50
F10000
\#CONTOUR MODE [MAX ANGLE=73]
N10 G01 X-P100 Y0 Z0 C0 A0
\$FOR P123 = 0, 90, 7.5
N2 G01 X0 Y0 Z0 C0 A0 G61
P1 = COS[P123]*P100
P2 = SIN[P123]*P100
NP123 XP1 YP2
N100 G01 X-P100 Y0 Z0 C0 A0
\$ENDFOR
\$FOR P123 = 270, 370, 7.5
N120 G01 X0 Y0 Z0 C0 A0 G61
P1 = COS[P123]*P100
P2 = SIN[P123]*P100
NP123 XP1 YP2
N400 G01 X-P100 Y0 Z0 C0 A0
\$ENDFOR

```
M30
\(\times 10^{5}\)
red alpha \(=43^{\circ}\), blue alpha \(=73^{\circ}\)

\(\times 10^{4}\)
red alpha \(=43^{\circ}\), blue alpha \(=73^{\circ}\)


\subsection*{11.2.6 Remarks}

If axes are released or fetched after programming G61 or G261/G260 (contouring at block end), the contouring mode cannot be executed.

\section*{Programing Example}

\section*{Contouring mode is not executed}
```

N10 G01 X100 Y0 Z0 F1000
N20 G01 X50 Y50 G61
N30 \#PUT AX [Z] (Contouring mode is not executed)
N40 G01 X100
N50 M30

```

With interim point contouring, the shape of the curve depends on the choice of interim point. The following curve shape is also possible:


\subsection*{11.3 Other processes}

\subsection*{11.3.1 Akima spline interpolation}

Notice
The use of this feature requires a license for the "Spline" extension package. It is not included in the scope of the standard license.

The target points of the programmed linear blocks (G00 and G01) are the vertices through which the Akima splines are fitted.

\subsection*{11.3.1.1 Selecting AKIMA spline type (\#SPLINE TYPE AKIMA )}


\section*{Release Note}

As of Build V2.11.2010.02 the command \#SPLINE TYPE AKIMA replaces the command \#SET SPLINETYPE AKIMA. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.

If provided in the system, the Akima spline is the default spline type. Nevertheless, it is recommended to specify the spline type explicitly.

Immediately after selecting the spline type, no tangential transition is possible from the preceding motion block to the spline curve.

\subsection*{11.3.1.2 Selecting Akima spline interpolation (\#SPLINE ON)}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#SPLINE ON replaces the command \#SET SPLINE ON. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

\section*{\#SPLINE ON \\ (modal)}

Akima spline interpolation is selected in the last mode selected. The Akima spline curve starts at the last target point programmed. This point is the first vertex of the Akima spline curve.

The command may be programmed in the same statement as the second vertex or in the preceding statement.

Alternatively, Akima spline interpolation can be selected using the G151 command.

\subsection*{11.3.1.3 Deselecting Akima spline interpolation (\#SPLINE OFF)}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#SPLINE OFF replaces the command \#SET SPLINE OFF. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.

Deselecting Akima spline interpolation is only permitted if at least 3 vertices were programmed or after 2 vertices if all tangents were specified (tangential transition at both ends of the curve or explicit programming of all tangents).

If the command is programmed in one statement together with a position, the respective point is not part of the Akima spline curve.
Alternatively, Akima spline interpolation can be selected using the G150 command.

\subsection*{11.3.1.4 Specifying transition type (spline curve) (\#AKIMA TRANS)}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#AKIMA TRANS [...] replaces the command \#SET ASPLINE MODE [...]. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.
\#AKIMA TRANS [ [ START<ident> END<ident> ] ]

START<ident> END<ident>

Types of tangential transition between the spline curve and the adjacent (linear or circular) motion blocks. Specification of the transition types is optional. If no specification is made, the AUTO presetting is used. Permitted identifiers:
AUTO Tangent of the spline curve at the transition is automatically calculated
TANGENTIAL Tangential transition to the preceding or succeeding block
USER Tangent of the spline curve at the transition is explicitly specified by the user


Fig. 90: Examples of combining transition types 1 and 2

If an explicit tangent specification is selected for a certain transition (transition type 3) and if no tangent is programmed for a certain axis, this tangent is automatically determined (transition type 1).

\subsection*{11.3.1.5 Defining the start tangent (\#AKIMA STARTVECTOR)}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#AKIMA STARTVECTOR... replaces the command \#SET ASPLINE STARTTANG.... For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.
\#AKIMA STARTVECTOR \{<axis_name><expr>\}
```

<axis_name><expr>

```

Components of the tangent vector, real value

Definition of start tangent using axis designations, a vector is specified whose direction corresponds to the direction of the tangent - the vector needs not therefore be normalised: However, all the components of the main axes must always be specified.

With tracking axes, the vector component is calculated from the ratio of the motion path of the tracking axis to the motion path of the contour:

Vector component \({ }_{\text {Mitschlepp }}=\mathrm{S}_{\text {Mitschlepp }} / \mathrm{S}_{\text {Bahn }}\) where \(\mathrm{S}_{\text {Bahn }}=\sqrt{S_{x}{ }^{2}+S_{y}{ }^{2}+S_{z}{ }^{2}}\)
\(\sqrt{S_{X}{ }^{2}+S_{y}{ }^{2}+S_{z}{ }^{2}}\)

\subsection*{11.3.1.6 Defining the end tangent (\#AKIMA ENDVECTOR)}
\(\left[\begin{array}{ll}\hline & \begin{array}{l}\text { Release Note } \\ \text { vone } \\ \text { 2060 } \\ 3300\end{array}\end{array} \begin{array}{l}\text { As of Build V2.11.2010.02 replaces the command \#AKIMA ENDVECTOR ... the command \#SET } \\ \text { ASPLINE ZIELTANG ...assigns the definition of the main spindle. For compatibility reasons, this } \\ \text { command is still available but it recommended not to use it in new NC programs. }\end{array}\right.\)
\#AKIMA ENDVECTOR \{<axis_name><expr>\} (modal)
<axis_name><expr> Components of the tangent vector, real value

Definition of target tangent; analogous to the definition of the start tangent.

\section*{Programing Example}

\section*{Definition of end tangent}


The following NC program supplies the same result but uses the second variant to select and deselect the spline interpolation.

N10 G01 X20 Y0 F1000
N20 \#AKIMA TRANS[START=USER END=AUTO] (Transition type with spec. of)
(start tangent and auto. determ)
(of end tangent)
N30 \# AKIMA STARTVECTOR X1 Y1 Z0 (Pre-set start tangent)
N40 G151 G01 X40 Y20
(Select spline interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70 (Deselect spline interpolation)
N100 M30

The following NC program supplies the same result but uses the second variant to select and deselect the spline interpolation.

N10 G01 X20 Y0 F1000
N20 \#AKIMA TRANS [START=USER END=AUTO] (Transition type with spec. of) (start tangent and auto. determ) (of end tangent)
N30 \# AKIMA STARTVECTOR X1 Y1 Z0 (Pre-set start tangent)
N40 G151 G01 X40 Y20 (Select spline interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70 (Deselect spline interpolation)
N100 M30

Caution: Block No. 80 contains the target point of the spline.

The program generates the following contour:


Fig. 91: Contour in the programming example (no. refers to 1st programming example)
It is clearly evident that the curve section corresponding to block N50 has the programmed slope 1 at its start (corresponding to the start of the spline curve). The slope at the spline end (end of block N90) is calculated automatically.

\section*{Notice}

If circular blocks (G02 or G03) are inserted, the spline curve is interrupted before the circular block and when the next linear block arrives, a new spline curve is started automatically. The transitions to and from the circular block are tangential.
The spline curve is also interrupted if a linear block is programmed with stationary main axes and moving tracking axes. Spline interpolation is deselected by the automatic determination of the tangent at curve end. Tracking axes are interpolated linearly until a linear block with moving main axes is programmed. If this is the case, spline interpolation is automatically reselected. The transition to the spline curve occurs tangentially for both the main and the tracking axes.

Other functions (e.g. M functions) may also be programmed between the linear blocks which serve as vertices. However, the count of these functions which were programmed between totally 5 consecutive vertices is restricted depending on the configuration.

\subsection*{11.3.2 B spline interpolation}

\section*{Notice}

The use of this feature requires a license for the "Spline" extension package. It is not included in the scope of the standard license.

The target points of the programmed linear blocks (G00 and G01) are the control point which are used to generate the B spline curve. It must be ensured that the B spline curve only runs through the control points only at the start and end.

\subsection*{11.3.2.1 Selecting B spline type (\#SPLINE TYPE BSPLINE)}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#SPLINE TYPE BSPLINE replaces the command \#SET SPLINETYPE BSPLINE. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.

If the control system only contains the B spline type, this spline type is selected automatically. Nevertheless, it is recommended to specify the spline type explicitly.

\subsection*{11.3.2.2 Selecting B spline interpolation (\#SPLINE ON)}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#SPLINE ON replaces the command \#SET SPLINE ON. For compatibility reasons, this command is still available but it's recommended not to use it in new NC programs.

\section*{\#SPLINE ON}

The B spline curve starts at the target point last programmed. The command may be programmed in the same statement as the second vertex or in the preceding statement.

Alternatively, B spline interpolation can be selected using the G151 command.

\subsection*{11.3.2.3 Deselecting B spline interpolation (\#SPLINE OFF)}

\section*{Release Note}

As of Build V2.11.2010.02 replaces the command \#SPLINE OFF the command \#SET SPLINE OFF. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.
```

\#SPLINE OFF (modal)

```

Deselecting B spline interpolation is only permitted if after least 4 vertices are programmed.
If the command is programmed in one statement together with a position, the respective point is no longer part of the \(B\) spline curve.

Alternatively, B spline interpolation can be deselected using the G150 command.

\section*{Programing Example}

Deselecting spline interpolation
```

N10 \#SPLINE TYPE BSPLINE
N20 G01 X0 Y50 Z0 F10000
N30 \#SPLINE ON
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 X50 Y37.5
N100 X55 Y32.5
N110 X58 Y12
N120 X70 Y12
N130 X77.5 Y10
N140 X90 Y35
N150 X100 Y37.5
N160 \#SPLINE OFF
N170 M30

```

The figure below shows the contour resulting from the programming example:


Fig. 92: Contour resulting from programming example

The figure shows the smoothing property of a B spline curve, especially for blocks N120/N130. Moreover, it shows that the curve does not pass through the control points. However, the polygon resulting from connecting the control points provides an estimation of the actual shape of the \(B\) spline curve.

\section*{Notice}

With a B spline curve, it is not possible to specify the tangents at curve ends directly. However, because the tangents of the \(B\) spline curve has the same tangents of corresponding motion blocks at the curve end, the tangents may be specified by programming the first and last motion blocks.

\subsection*{11.3.3 PSC programming with OP1 and OP2}
\#HSC [ON | OFF] [ [OPMODE <expr>] [CONTERROR <expr>] ] (modal)
\(\left.\begin{array}{l}\text { ON } \\
\text { OFF } \\
\text { OPMODE <expr> }\end{array} \begin{array}{l}\text { Enable HSC programming } \\
\text { Disable HSC programming } \\
\text { Set operation mode: }\end{array}\right]\)\begin{tabular}{|l|l|}
\hline Valid values & Meaning \\
\hline \(\mathbf{1}\) & Insert transition polynomials \\
\hline \(\mathbf{2}\) & Generate interpolating spline curves \\
\hline
\end{tabular}

CONTERROR <expr> Define maximum contour error in [mm, inch]:
\begin{tabular}{|l|l|}
\hline Valid values & Meaning \\
\hline\(>=\mathbf{0 . 0}\) & \begin{tabular}{l} 
Maximum contour error " \(\varepsilon "\) \\
Is only valid in combination with active OPMODE 1. \\
If the parameter is set in OPMODE 2, it only is effective after chan- \\
ging to OPMODE 1.
\end{tabular} \\
\hline
\end{tabular}

If no parameters are programmed when HSC programming is enabled, the following default values are valid:
\begin{tabular}{|l|l|}
\hline OPMODE & 1 \\
\hline CONTERROR & 0.1 mm \\
\hline
\end{tabular}

\section*{Notice}

The parameters may also be specified in several steps. This means that it is possible to first enable the operation mode ("OPMODE") and HSC processing ("ON") in the first command and then change the maximum contour error ("CONTERROR") in a second command.
An error message is generated if the operation mode is changed while HSC processing is enabled.

\subsection*{11.3.3.1 Available operation modes}

\section*{OPMODE 1: Insert transition polynomials}

In operation mode 1, the blocks are shortened at motion block transitions and transition polynomials are inserted.


Fig. 93: Insert transition polynomials
The maximum contour error CONTERROR (" \(\varepsilon\) ") is required to generate the polynomial transition. The inserted polynomial reduces the actual contour error (i.e. \(\varepsilon\) real \(\leq \varepsilon p r o g\) ).

\section*{Notice}

No polynomial is inserted in the current block transition if
- the block length of the first block is \(<1.1 \mu \mathrm{~m}\) or
- the block length of the target block is \(<2.2 \mu \mathrm{~m}\) or
- the knee angle is \(>178^{\circ}\).

\section*{OPMODE 2: Generating spline curves for HSC programming}

In operation mode 2, spline curves are generated by the specified edge points. At the start of prismatic parts, spline generation is automatically deselected and a tangential transition is executed.

With block transitions in the prismatic range which are detected based on the additional parameters as described in Section PSC programming with OP1 and OP2 [ 300] an automatic change takes place to operation mode 1 (OPMODE 1), i.e. transition polynomials are inserted. Contour deviation between vertices is not monitored for inserted spline curves.


Fig. 94: Generating spline curves for HSC programming

Programing Example
OPMODE 1: Insert transition polynomials
```

\#HSC[OPMODE 1 CONTERROR 0.01] (Operation mode 1)
(Max. contour error <= 0.01 mm)

```
\#HSC ON
\#HSC OFF
\#HSC ON
. .
\#HSC OFF
\#HSC ON [OPMODE 2 CONTERROR 0.002] (Enable free-form programming) (With operation mode 2)
(Max. contour error \(<=0, .02 \mathrm{~mm}\) )
\#HSC[CONTERROR 0.005] (Change ma. cont. error \(<=0.005 \mathrm{~mm}\) )
\#HSC OFF

\section*{Programing Example}

OPMODE 2: Generating spline curves for HSC programming
```

\#HSC[OPMODE 1 CONTERROR 0.01] (Select operation mode 1)
(Max. contour error <= 0.01 mm)
(Enable HSC programming)
(Error message!)
(Changing operation mode is)
(not allowed while HSC programming)
(is active)

```

\subsection*{11.3.3.2 Additional parameters}

Normally, only the previously described parameters are required. For access to internal settings for operation mode 2, the following additional parameters are available:
\#HSC [ [COS_PHI_MIN<expr>] [FACT_BLOCK_LEN<expr>]]
\begin{tabular}{|l|l|l|l|}
\hline HSC parameters & Type & \begin{tabular}{l} 
Valid range \\
(Default)
\end{tabular} & Description \\
\hline COS_PHI_MIN & Real & \begin{tabular}{l}
\(-1.0 \ldots 1.0\) \\
\((0.7)\)
\end{tabular} & \multicolumn{1}{|c|}{ Block i } \\
& Real & \begin{tabular}{l}
\(>0.0\) \\
\((3.0)\)
\end{tabular} & \begin{tabular}{l} 
Minimum value for cos \((\varphi)\). Smaller values lead to \\
deselection of spline curve generation.
\end{tabular} \\
\hline FACT_BLOCK_LEN & \begin{tabular}{l} 
Maximum factor by which the block length may exceed \\
the median block length *. Larger values lead to deselec- \\
tion of spline curve generation. \\
\(*\) The median block length results from the block lengths \\
of the previous 5 motion blocks.
\end{tabular} \\
\hline
\end{tabular}

\section*{12 Additional functions}

A complete list of additional functions is contained in the overview of commands in the Appendix under Additional functions [ 809] (\#..).

Additional functions are a separate group of NC text commands. The permit the programming of specific extensions and technological processes which are covered by DIN/ISO programming. The syntax for additional functions is:
```

\#<string> <spezifische Zusatzsyntax>

```
\#<string>
<specific additional syntax>

Plaintext command. Between \# and <string> no blanks are permitted.
Subsequent command-specific syntax elements which are programmed as additional strings directly or within brackets.

\begin{abstract}
Attention
Each \# command must be configured alone in a separate NC line. Any exceptions are specifically pointed out.
\end{abstract}

\section*{Notice}

If not otherwise displayed explicitly, commas "," and equals signs "=" are optional in the specific additional syntax and are only used to improve the legibility of the NC program.

\section*{Example:}
\#STRING [A_VALUE 10 B_VALUE 20] \(\longrightarrow\) \#STRING [A_VALUE =10, B_VALUE=20]

\subsection*{12.1 Restoring axis configurations and axis couplings}

Complex multi-channel machines which use axis exchange operations and synchronous axes (synchronous mode) for machining are aborted by an NC reset if an error occurs. The NC channels are then reinitialised and the channel configurations valid at the time of the abort are lost. To permit clearance of tool axes, it is necessary to restore the axis configuration which was active at the time of abort and comprising the axes found in the channel and the selected axis couplings. This is implemented by the NC commands described below.

\subsection*{12.1.1 Saving a current configuration (\#SAVE CONFIG)}
\#SAVE CONFIG [ [ AX ] [ AXLINK ] ] (modal)

AX: \(\quad\) Save the current axis configuration in the NC channel.
AXLINK: Save the current axis couplings selected in the NC channel. Couplings that were previously saved are deleted when AXLINK is programmed although no couplings are active.

The command \#SAVE CONFIG is usually always used in the NC program after operations which cause a change in axis configuration or in the activation of axis couplings. For example, after an axis exchange command or after a command to select synchronous mode.

During the save operation, configuration data is sent from the decoder over the NC channel to the interpolator. The interpolator then sends this data directly back to the decoder where it is saved internally. This operation ensures that the current configuration of axes and axis couplings is stored in the decoder synchronously with the current processing state.

\section*{Programing Example}

\section*{Saving a current configuration}
```

%main
N10 X0 Y0 Z0
N15 \#AX REQUEST [C,4,5] [B,5,6]
N20 \#AX LINK [1, C=X, B=Y]
N25 \#AX LINK [2, B=X]
N30 \#AX LINK ON [1]
N35 \#SAVE CONFIG [AX AXLINK] Save axis configuration X,Y,Z,B,C
and the active coupling group 1
N.. X.. Y.. Z..
N200 \#AX LINK OFF ALL
N210 \#AX RELEASE [C]
N220 \#SAVE CONFIG [AX] Save new axis configuration X,Y,Z,B.
The previously saved axis configuration
from block N35 is overwritten!
N.. X.. Y.. Z..
N99 M30

```

\section*{Notice}

A saved configuration remains stored program global. It can only be overwritten by subsequent \#SAVE CONFIG commands or deleted by \#CLEAR CONFIG.

\subsection*{12.1.2 Loading or restoring a saved configuration (\#LOAD CONFIG)}
```

\#LOAD CONFIG [[ AX ][ AXLINK ]] moda

```

AX: Loading the last axis configuration saved. If no saved configuration is found, an error message is output. Axes which are not found in the NC channel are requested without axis offsets.
AXLINK: Loading the last active axis couplings saved. All axis couplings are summarised, restored and activated in coupling group 1. If no saved axis couplings are found, an error message is output.

The command \#LOAD CONFIG is best used after an NC RESET in the clearance program to restore the last configuration saved. The machine operator is responsible for saving the required configuration in the main program by using \#SAVE CONFIG and for correctly restoring the required configuration. If both keywords are programmed, the axis configuration is always first restored completely in the NC channel (without "FAST") irrespective of sequence in which the keywords are programmed. Then the last synchronous mode saved is re-activated.

Programing Example

\section*{Loading or restoring a saved configuration}

Start clearance program after abort of processing and NC RESET:
```

%Clearance
N10 G53
N35 \#LOAD CONFIG [AX AXLINK] Restore the saved axis
configuration and axis couplings
under coupling group 1
N40 \#ECS ON Define an ECS to execute
withdrawal strategy
N.. X.. Y.. Z..
N200 \#AX LINK OFF ALL
:
N.. X.. Y.. Z..
N99 M30

```

\section*{Notice}

When \#SAVE CONFIG and \#LOAD CONFIG are used in the same NC program, an implicit FLUSH is always executed at the start of \#LOAD CONFIG to ensure consistency of the saved configurations in the channel.

\section*{Programing Example}

NXX \#SAVE CONFIG [AX AXLINK]
N. .

Nxx \#LOAD CONFIG [AX AXLINK]
First, implicit FLUSH then restore axis configuration, then restore axis couplings
N. . X. Y. . Z..

N99 M30

\subsection*{12.1.3 Clearing a current configuration (\#CLEAR CONFIG)}
\#CLEAR CONFIG

The command \#CLEAR CONFIG completely clears the last configuration saved. A \#LOAD CONFIG programmed directly after causes the error message "No restorable configuration found". This means that \#SAVE CONFIG must first be used to save a configuration again before it can be restored with \#LOAD CONFIG.

For the machine operator, a \#CLEAR CONFIG is always helpful if he wants to prevent access to a configuration that may have been incorrectly saved to different NC programs.

\section*{Programing Example}

Deleting a saved configuration

Execute clearance program, then delete the saved configuration:
```

%Clearance
N10 G53
N35 \#LOAD CONFIG [AX AXLINK] Restore the saved axis
configuration and axis couplings
under coupling group I
N40 \#ECS ON Define an ECS to execute
withdrawal strategy
N.. X.. Y.. Z..
:
N200 \#AX LINK OFF ALI
:
N.. X.. Y.. Z..
N.. \#CLEAR CONFIG Clear saved configuration after
the withdrawal motion ends
N99 M30

```

\subsection*{12.2 Axis exchange commands}

This section describes NC commands to
- Request axes
- Release axes and to
- Define an axis configuration

At every program start, the axis configuration specified in the channel parameter list [1] [> 819]-5 is restored. Axis exchange commands are active in the currently selected axis group. An NC block may contain several axis requests and/or returns. These operations are executed in semi-parallel state.

\section*{Notice}

It is not permitted to replace axes for which synchronous or manual mode is active.
When TRC is active, the first two main axes may not be replaced.
Axis exchange commands must be programmed alone in a separate NC line.
Axis exchange commands are only valid for path and tracking axes. Spindle axes are ignored. There are special commands [ \(>677\) ] provided to release and request spindle axes.

As of Version V2.6, the previously available axis exchange commands were revised and extended in functionality and mode of operation. The Section "Standard axis exchange commands" describes the standard syntax available until Version V2.6. This syntax may continue to be used in the future.

The Section "Extended axis exchange commands" describes the new syntax. The syntax is fully downwards compatible with the previous scope of functions. However, the logic switches and additional functions to define axis exchange sequences offer more flexible programming options.

\subsection*{12.2.1 Standard syntax}

\subsection*{12.2.1.1 Requesting axes (\#CALL AX)}

\section*{Notice}

If an axis which is already present in the axis group of the NC channel is requested, no request is triggered for this axis.

The following NC command requests axes from the axis management.
\#CALL AX [<Modus>] [<Achsname>,<Achsnummer>,<Achsindex> \{,<Optionen>\} ]
\(\{[<A c h s n a m e>,<A c h s n u m m e r>,<A c h s i n d e x>\{,<\) Optionen \(>\}]\}\) (non-modal)
<mode > With/without request for axis positions from the interpolator and a position initialisation of the NC channel when axes are replaced.
\begin{tabular}{|l|l|}
\hline Mode & Requesting an axis in the NC channel \\
\hline & \begin{tabular}{l} 
Default setting: With the request for set values from the interpolator and a posi- \\
tion initialisation of the NC channel,
\end{tabular} \\
\hline FAST & \begin{tabular}{l} 
Without request for set values from the interpolator. Position initialisation of the \\
NC channel.
\end{tabular} \\
\hline
\end{tabular}

Achsname> The permissible strings for axis designation start with the letters \(A, B, C, Q, U, V, W\), \(\mathrm{X}, \mathrm{Y}\) and Z . The multiple assignment of the same designation for several axes (identification by logical axis number) generates an error message and the NC program is aborted.
<Achsnummer>
<Achsindex>

The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known in the axis management. If an unknown logical axis number or several identical logical axis numbers are requested, it results in an error message and the NC program is aborted.
The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1 . The axis index may not yet be assigned an axis. If a request is made for an index that is already assigned a different axis, it results in an error message and the NC program is aborted.
\begin{tabular}{|l|l|}
\hline Index & Axis configuration \\
\hline \(\mathbf{0}\) & 1st main axis in the machining plane. \\
\hline \(\mathbf{1}\) & 2nd main axis in the machining plane. \\
\hline \(\mathbf{2}\) & 3rd main axis generally perpendicular to the machining plane. \\
\hline \(\mathbf{3}\) & 1. Tracking axis. \\
\hline\(\ldots \mathbf{n}\) & (n-2). Tracking axis. \\
\hline
\end{tabular}

\section*{Attention}

To simplify programming, it is possible to leave the specification for the axis index for tracking axes empty (with \#CALL AX... only). In this case, the next free axis index after index 3 is then assigned automatically to this tracking axis.
For main axes the index must always be specified explicitly.
However, it must be noted that the index of a tracking axis is important with regard to various functionalities. For example, all transformation axes must be arranged with no gaps after the main axes in the case of kinematic transformation (RTCP). In such cases, it is then necessary to program the axis index explicitly for the tracking axes.

Offsets are kept axis-specific. This refers to the following offsets:
- Reference point offset
- Clamping offset
- Tool offset
- Zero offset
- Measuring offset
- Manual mode offset
- Position preset

Offset adoption for requesting axes can be controlled using the keywords from the table below.

\section*{Notice}

With exception of tool offsets, all other offsets are always attached to the logical axis number an axis release or request (see programming example "Zero_offset" below).
\begin{tabular}{|l|l|}
\hline Keyword & Request axes and \\
\hline & No adoption of offsets (default) \\
\hline ALL & Adopt all offsets * \\
\hline BPV & Adopt reference point offset \\
\hline PZV & Adopt clamping offset \\
\hline WZV & Adopt tool offset * \\
\hline NPV & Adopt zero offset \\
\hline MOFFS & Adopt measuring offset \\
\hline SOFFS & Adopt manual mode offset \\
\hline PSET & Adopt position preset \\
\hline
\end{tabular}

\begin{abstract}
Attention
* With \#CALL AX, adoption of tool offsets only makes sense when the tool is deselected. As soon as a tool is active in the channel or if it is selected with \#CALL AX after replacement, the adopted tool offsets are replaced with the offsets of the current tool.

It is therefore recommended to execute \#CALL AX when a tool is deselected.
When a tool is selected, please note that the offsets are always included in the axis calculations depending on the sequence indexed in the tool data.
\end{abstract}

\section*{Programing Example}

\section*{Requesting axes}
```

%Zero_Offset
N010 X200
N015 G54
N015 V.G.NP_AKT.V.X = 11
N016 X0 (Machine X axis end position is 11)
NO20 \#PUT AX [X] (Release axis X with log. axis no. 1)
N130 \#CALL AX [X1,1,0,NPV]
N140 X1=100

```
M30

\section*{Example:}

Assign axis names, logical axis numbers and axis indices at program start:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{\%Achstausch1}

N10 \#CALL AX FAST [X1,7,4] (X1 axis without request for command values) (and output of init. function block)

Assign axis names, logical axis numbers and axis indices after the axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline \(\mathbf{X}\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline X1 & 7 & 3 \\
\hline
\end{tabular}

\section*{Programming example continued:}
:
N100 \#CALL AX \([Y 1,8,6][C, 9\),\(] \quad (Request Y 1\) and \(C\) axes, axis index is automatically defined by

C axis)

Assign axis names, logical axis numbers and axis indices after second axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline X & 1 & 0 \\
\hline Y & 2 & 1 \\
\hline Z & 3 & 2 \\
\hline C & 9 & 3 \\
\hline X1 & 7 & 4 \\
\hline Y1 & 8 & 5 \\
\hline
\end{tabular}

\section*{Programming example continued:}
N1000 \#CALL AX FAST [Z1,13,5,ALL] (Adopt all offsets)

N1010 \#CALL AX [C1,11,7,NPV MOFFS] (Adopt zero)
(and measuring offsets)

Assign axis names, logical axis numbers and axis indices after third axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline X & 1 & 0 \\
\hline Y & 2 & 1 \\
\hline Z & 3 & 2 \\
\hline C & 9 & 3 \\
\hline Z1 & 7 & 4 \\
\hline Y1 & 13 & 5 \\
\hline C1 & 8 & 6 \\
\hline
\end{tabular}

\subsection*{12.2.1.2 Releasing axes (\#PUT AX, \#PUT AX ALL)}

This NC command returns axes of the axis group of the NC channel to the axis management. It is permitted to return axes which are not or no longer present and this generates no error message.
```

\#PUT AX [ <Achsname> {,<Achsname> } ] (non-modal)

```
<Achsname>
Axis designations may consist of strings with the starting characters \(\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{Q}, \mathrm{U}, \mathrm{V}\), \(W, X, Y\) and \(Z\).

This NC command returns all the axes in the axis group of the NC channel to the axis management.

\section*{\#PUT AX ALL}
(non-modal)

\section*{Example}


Assigning axis names, logical axis numbers and axis indices at program start:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

\section*{Programing Example}

Releasing axes
```

N10 \#PUT AX [ X, B] (Release X axis; B axis not present)
(No error message is output)

```

Assign axis names, logical axis numbers and axis indices after axis release:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline & & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

\section*{Programming example continued:}

N100 \#PUT AX ALL (Release all axes of this group.)
Assign axis names, logical axis numbers and axis indices after second axis release:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline & & 0 \\
\hline & & 1 \\
\hline & & 2 \\
\hline
\end{tabular}
kernel Industrielle Steuerungstechnik GmbH

\subsection*{12.2.1.3 Definition of an axis configuration (\#SET AX)}

This NC command defines a new axis configuration to replace the existing axis configuration. Only exactly the axes which are programmed in NC command form the new axis configuration of the NC channel.
```

\#SET AX [<Modus>] [<Achsname>,<Achsnummer>,<Achsindex> {,<Optionen>}]
{ [<Achsname>,<Achsnummer>,<Achsindex>{,<Optionen>}]} (non-modal)

```
<Modus> With/without request for axis positions from the interpolator and a position initialisation of the NC channel when axes are replaced.
\begin{tabular}{|l|l|}
\hline Mode & Replacing axes in the NC channel \\
\hline & \begin{tabular}{l} 
Default setting: With request for setpoint values from the interpolator and a po- \\
sition initialisation of the NC channel
\end{tabular} \\
\hline FAST & \begin{tabular}{l} 
Without request for set values from the interpolator. Position initialisation of the \\
NC channel.
\end{tabular} \\
\hline
\end{tabular}
<Achsname>
<Achsnummer>
<Achsindex>

The permissible strings for axis designation start with the letters \(A, B, C, Q, U, V, W\), \(\mathrm{X}, \mathrm{Y}\) and Z . The multiple assignment of the same designation for several axes (identification by logical axis number) generates an error message and the NC program is aborted.

The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known by the axis management. If an request is made for an unknown logical axis number or several identical logical axis numbers, an error message is output and the NC program is aborted.

The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis. If a request is made for an index that is already assigned a different axis, it results in an error message and the NC program is aborted.
\begin{tabular}{|l|l|}
\hline Index & Axis configuration \\
\hline \(\mathbf{0}\) & 1. Main axis in the machining plane. \\
\hline \(\mathbf{1}\) & 2nd Main axis in the machining plane. \\
\hline \(\mathbf{2}\) & 3rd Main axis generally perpendicular to the machining plane. \\
\hline \(\mathbf{3}\) & 1. Tracking axis. \\
\hline\(\ldots \mathbf{n}\) & (n-2). Tracking axis. \\
\hline
\end{tabular}

Offsets are kept axis-specific. This refers to the following offsets:
- Reference point offset
- Clamping offset
- Tool offset
- Zero offset
- Measuring offset
- Manual mode offset
- Position preset

Offset adoption for requesting axes can be controlled using the keywords from the table below.
\begin{tabular}{|l|l|}
\hline Keyword & Replacing axes and \\
\hline & No adoption of offsets (default) \\
\hline ALL & Adopting all offsets * \\
\hline BPV & Adopting reference point offset \\
\hline PZV & Adopting clamping offset \\
\hline WZV & Adopting tool offset * \\
\hline NPV & Adopting zero offset \\
\hline MOFFS & Adopting measuring offset \\
\hline SOFFS & Adopting manual mode offset \\
\hline PSET & Adopting position preset \\
\hline
\end{tabular}

\section*{Attention}
* When the tool is selected, please note the following when adopting tool offsets with \#SET AX:
- If axes are only swapped (internal axis replacement) by \#SET AX and otherwise no additional axes are specified or requested, all offsets (including the tool offsets) are also replaced and continue to remain active. Specifying keywords to adopt offsets has no effect. If a new tool is then selected, the offsets also exchanged are replaced by the offsets of the new tool.
- As soon as an axis release or an axis request is triggered by \#SET AX (external axis exchange), tool offsets are again included in the calculation of the sequence of the axes indexed in the tool data. Therefore, any adopted tool offsets are replaced by the current tool offsets. If the original tool offsets in the corresponding axes should continue to apply, a new tool must be selected with offsets which have been adapted to the new axis arrangement.

You are therefore advised to run \#SET AX when a tool is deselected to ensure the correct assignment of tool offsets by the appropriate parameterisation in the data record of a new tool selected.

\section*{Example:}
\begin{tabular}{|l|c|c|c|c|}
\hline Index of tool offsets in tool data & {\([0]\)} & {\([1]\)} & {\([2]\)} & [3] \\
\hline Parameterised tool offsets e.g. for T1 & 50 & 0 & 70 & 20 \\
\hline Axis configuration at program start & X & Y & \(\mathbf{Z}\) & --- \\
\hline Included tool offsets after T1 selection & 50 & 0 & 70 & --- \\
\hline "Internal" \#SET AX \{Z, X, Y\}: & \(\mathbf{Z}\) & \(\mathbf{X}\) & \(\mathbf{Y}\) & --- \\
\hline Tool offsets are also swapped or & 70 & 50 & 0 & --- \\
\hline "External" \#SET AX \{Z, X, Y, B\}: & \(\mathbf{Z}\) & \(\mathbf{X}\) & \(\mathbf{Y}\) & \(\mathbf{B}\) \\
\hline Tool offsets are recalculated corresponding to tool T1 & 50 & 0 & 70 & 20 \\
\hline
\end{tabular}

\section*{Example:}

Assigning axis names, logical axis numbers and axis indices at program start:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

\section*{Programing Example}

Set axis configuration:
```

(X axis remains in place;)
(Y axis is released;)
(Z axis is re-sorted acc. to Index 4;)
(Y1 and Z1 axis are requested)
%ACHSTAUSCH1
N10 \#SET AX [X,1,0][Y1,4,2][Z1,5,3][Z, 3, 4]

```

Assign axis name, logical axis numbers and axis indices after N10:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline X & 1 & 0 \\
\hline Y 1 & & 1 \\
\hline Z1 & 4 & 2 \\
\hline Z & 5 & 3 \\
\hline
\end{tabular}

\subsection*{12.2.2 Extended syntax}

\section*{Release Note}

\section*{The availability of this function depends on the configuration and concrete scope of version.}

Extended syntax permits the programming of axis exchange sequences with macro definitions (Section macros [ 709]) or external variables of the string type (V.E....). This is especially useful for multi-channel machines and systems when static axis groups should be replaced between channels. For example, these axis groups can be defined in macros and used in axis exchange commands.

In addition, the extended syntax permits the internal processing of conflicts by setting so-called logic switches without the output of an error message or a warning. Logic switches are optional and can be programmed additionally in the command. If no logic switches are programmed, standard evaluation applies as before. This means that, if there are plausibility conflicts, the evaluation of axis exchange sequences is aborted and error messages are output.

The logic evaluation is identical for all axis exchange commands and also checks for plausibility in the programmed axis exchange sequence and plausibility to existing axes in the NC channel.

Logic switches are especially useful for axis exchange sequences which are defined by macros or string variables because any overlaps occurring here and redundant programmings can be resolved internally.

\subsection*{12.2.2.1 Requesting axes (\#AX REQUEST)}

The following NC command requests axes from the axis management.
```

\#AX REQUEST [NAM, NBR, IDX] [<axis_exchange_sequence> {,<options>}]
{ [<axis exchange sequence> {,<options>} ] } (non-modal)

```

NAM, NBR, IDX Logic switches for handling conflicts.
\begin{tabular}{|l|l|}
\hline ID & Meaning \\
\hline NAM & Handling redundant axis names \\
\hline NBR & Handling redundant axis numbers \\
\hline IDX & Handling redundant axis indices \\
\hline
\end{tabular}

\section*{Notice}

Logic switches can be programmed individually or in combination.
<Achstauschsequenz>
<Achsname>
consisting of:
The permissible strings for axis designation start with the letters \(A, B, C, Q, U, V, W, X\), \(Y\) and \(Z\).

\section*{Notice}

In case of conflicts within the programmed axis exchange sequence:
Redundant axis names \(\rightarrow\) ERROR, program abort
In case of conflicts with existing axes in the NC channel:
Identical axis names, different axis numbers \(\rightarrow\) ERROR, program abort
If the logic switch NAM is set, the conflict is cleared as follows:
The axis receives the default axis name from its axis parameter list P-AXIS-00297. The user must ensure a clear definition of the default axis name in the list.

The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known in the axis management.

Notice
In case of conflicts within the programmed axis exchange sequence:
Redundant axis numbers \(\rightarrow\) ERROR, program abort
In case of conflicts with existing axes in the NC channel:
Axis number already exists in NC channel \(\rightarrow\) WARNING
If the logic switch NBR is set, the conflict is cleared as follows:
The axis request is ignored, i.e. it is not executed.

\section*{<Achsindex>}

The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis.
\begin{tabular}{|l|l|}
\hline Index & Axis configuration \\
\hline \(\mathbf{0}\) & 1. Main axis in the machining plane. \\
\hline \(\mathbf{1}\) & 2nd Main axis in the machining plane. \\
\hline \(\mathbf{2}\) & 3rd Main axis generally perpendicular to the machining plane. \\
\hline \(\mathbf{3}\) & 1. Tracking axis. \\
\hline\(\ldots \mathbf{n}\) & (n-2). Tracking axis. \\
\hline
\end{tabular}

\begin{abstract}
Attention
To simplify programming, it is possible to leave the specification for the axis index for tracking axes empty. In this case, the next free axis index after index 3 is then assigned automatically to this tracking axis.

For main axes the index must always be specified explicitly.
However, it must be noted that the index of a tracking axis is important with regard to various functionalities. For example, all transformational axes must be arranged after the main axes with no gaps in the case of kinematic transformation (RTCP). In such cases, it is then necessary to program the axis index explicitly for the tracking axes.
\end{abstract}

\section*{Notice}

In case of conflicts within the programmed axis exchange sequence:
Redundant axis numbers \(\rightarrow\) ERROR, program abort
In case of conflicts with existing axes in the NC channel:
Axis index is already assigned in the NC channel, different axis \(\rightarrow\) ERROR, program abort. With set logic switch IDX is set, the conflict is cleared as follows:
The next free index in the axis configuration of the NC channel is automatically determined for the axis.

\section*{<options>}

Offsets are kept axis-specific. This refers to the following offsets:
- Reference point offset
- Clamping offset
- Tool offset
- Zero offset
- Measuring offset
- Manual mode offset
- Position preset

Offset adoption for requesting axes can be controlled using the keywords from the table below.

\section*{Notice}

With the exception of tool offsets, all other offsets are always attached to the logical axis number on axis release or request.
\begin{tabular}{|l|l|}
\hline Keyword & Requesting axes and \\
\hline & no adoption of offsets (default) \\
\hline ALL & Adopting all offsets * \\
\hline BPV & Adopting reference point offset \\
\hline PZV & Adopting clamping offset \\
\hline WZV & Adopting tool offset * \\
\hline NPV & Adopting zero offset \\
\hline MOFFS & Adopting measuring offset \\
\hline SOFFS & Adopting manual mode offset \\
\hline PSET & \\
\hline
\end{tabular}

\section*{Attention}
* In the case of \#AX REQUEST, it only makes sense to adopt tool offsets when the tool is deselected. As soon as a tool is active in the channel or if it is selected with \#AX REQUEST after replacement, the adopted tool offsets are replaced with the offsets of the current tool.
It is therefore recommended to execute \#CALL AX when a tool is deselected.
When a tool is selected, please note that the offsets are always included in the axis calculations depending on the sequence indexed in the tool data.

\section*{Example}

\section*{Using standard functionality:}

Assigning axis names, logical axis numbers and axis indices at program start:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Requesting axes}

N10 \# AX REQUEST [X1,7,4] ;Request X1 axis

Assign axis names, logical axis numbers and axis indices after an axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline \(\mathbf{X}\) & 1 & 0 \\
\hline \(\mathbf{Y}\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline & & 3 \\
\hline\(X 1\) & 7 & 4 \\
\hline
\end{tabular}

\section*{Programming example continued:}
```

N100 \#AX REQUEST [Y1,8,6] [C,9, ] ;Request Y1 and C axis,
; C axis is set automatically
; to index 3

```

Assigning axis names, logical axis numbers and axis indices after second axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline C & 9 & 3 \\
\hline X1 & 7 & 4 \\
\hline Y1 & 8 & 5 \\
\hline
\end{tabular}

\section*{Programming example continued:}
```

N1000 AX REQUEST FAST [Z1,13,5, ALL] ;Adopt all offsets
N1010 AX REQUEST [C1,11,7, NPV MOFFS] ;Adopt zero
;offset and measuring offset

```

Assigning axis names, logical axis numbers and axis indices after third axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline X & 1 & 0 \\
\hline Y & 2 & 1 \\
\hline Z & 3 & 2 \\
\hline C & 9 & 3 \\
\hline X1 & 7 & 4 \\
\hline Z1 & 13 & 5 \\
\hline C1 & 8 & 6 \\
\hline
\end{tabular}

\section*{Example 2:}

Use the additional syntax (logic switches):
Assigning axis names, logical axis numbers and axis indices at program start in channel 1:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

Assigning axis names, logical axis numbers and axis indices at program start in channel 2 :
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 7 & 0 \\
\hline\(Y\) & 8 & 1 \\
\hline\(Z\) & 9 & 2 \\
\hline
\end{tabular}

\section*{Programing Example}

In channel 1: Request \(X\) and \(Y\) axes from channel 2 (logical axis numbers 7 and 8 ).

N10 \#AX REQUEST NAM [X,7,3] [Y,8,4] ;Request X/Y axes

Due to the logic switch NAM, the new additional axes \(X\) and \(Y\) in channel 1 are assigned their default names in the axis lists (e.g. X2 and Y2).
Assigning axis names, logical axis numbers and axis indices after axis requests:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline \(\mathbf{X}\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline\(X 2\) & 7 & 3 \\
\hline\(Y 2\) & 8 & 4 \\
\hline
\end{tabular}

\section*{Programming example continued:}

In channel 1: Request \(Z\) axis from channel 2 (logical axis number 9).

N100 \#AX REQUEST NAM IDX [Z,9,2] ;Request Z axis

Due to the logic switches NAM and IDX, the new additional \(Z\) axis in channel 1 is adopted in the next free index (e.g. index 5) together with its default name in the axis list (e.g. Z2).
Assigning axis names, logical axis numbers and axis indices after second axis request:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline\(X 2\) & 7 & 3 \\
\hline\(Y 2\) & 8 & 4 \\
\hline\(Z 2\) & 9 & 5 \\
\hline
\end{tabular}

\section*{Example 3:}

Use the additional syntax (logic switches and axis exchange sequences in string format):
Assigning axis names, logical axis numbers and axis indices at program start in channel 1 :
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

Assigning axis names, logical axis numbers and axis indices at program start in channel 2:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline X_1 & 7 & 0 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline Y & 8 & 1 \\
\hline\(Z\) & 9 & 2 \\
\hline A & 10 & 3 \\
\hline B & 11 & 4 \\
\hline
\end{tabular}

\section*{Programing Example}

In channel 1: Request \(X\) _1/ \(Y\) and \(B\) axis from channel 2 (logical axis numbers 7, 8 and 11). The axis exchange sequence is stored in a macro.
```

N05 "ACHSEN_KANAL2" = "[X_1,7,0] [Y,8,1] [B,11,2]"
:
N10 \# AX REQUEST NAM IDX "ACHSEN KANAL2" ; Request axes

```

Due to the logic switches NAM and IDX, the new additional axes in channel 1 are adopted correctly.

Assigning axis names, logical axis numbers and axis indices after axis requests:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline X_1 & 7 & 3 \\
\hline Y2 & 8 & 4 \\
\hline B & 11 & 5 \\
\hline
\end{tabular}

\subsection*{12.2.2.2 Releasing axes (\#AX RELEASE, \#AX RELEASE ALL)}

These NC command axes return axes in the axis group of the NC channel to the axis management. It is permitted to return axes which are not or no longer present and this generates no error message.
```

\#AX RELEASE [ <Achsname> {,<Achsname> } ]
(non-modal)

```
<Achsname> Axis names of the axes currently present in NC channel.

The logic switch NBR switches over evaluation from axis names to logical axis numbers (e.g. if axis names are unknown at the time of release).
```

\#AX RELEASE [NBR] [ <expr> {,<expr> } ] (non-modal)

```

This NC command returns all the axes in the axis group of the NC channel to the axis management.
\#AX RELEASE ALL

\section*{Example}

Assigning axis names, logical axis numbers and axis indices at program start:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline X & 1 & 0 \\
\hline Y & 2 & 1 \\
\hline Z & 3 & 2 \\
\hline B & 4 & 3 \\
\hline & 5 & 4 \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Releasing axes}

N10 \#AX RELEASE[X, A] ;Release X/A axes

Assigning axis names, logical axis numbers and axis indices after axis release:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline & & \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline B & 5 & 4 \\
\hline
\end{tabular}

Programming example continued:

N100 \#AX RELEASE NBR[2] ;Release Y axis

Assigning axis names, logical axis numbers and axis indices after second axis release:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline & & \\
\hline & & \\
\hline Z & 3 & 2 \\
\hline B & 5 & 4 \\
\hline
\end{tabular}

Programming example continued:

N100 \#AX RELEASE ALL ;Release all existing axes of this ; channel

Assigning axis names, logical axis numbers and axis indices after the third axis release:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline & & \\
\hline & & \\
\hline & & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|l|}
\hline & & \\
\hline & & \\
\hline
\end{tabular}

\subsection*{12.2.2.3 Definition of an axis configuration (\#AX DEF)}

This NC command defines a new axis configuration to replace the existing axis configuration. Only exactly the axes which are programmed in NC command form the new axis configuration of the NC channel.
\#AX DEF [NAM, NBR, IDX] [<axis_exchange_sequence> \{,<options>\}]
\{[<axis_exchange_sequence> \{,<options>\}] \} (non-modal)

NAM, NBR, IDX Logic switches for handling conflicts.
\begin{tabular}{|l|l|}
\hline ID & Meaning \\
\hline NAM & Handling redundant axis names \\
\hline NBR & Handling redundant axis numbers \\
\hline IDX & Handling redundant axis indices \\
\hline
\end{tabular}

\section*{Notice}

Logic switches can be programmed individually or in combination.
<Achsname>
consisting of:
The permissible strings for axis designation start with the letters \(A, B, C, Q, U, V, W, X\), \(Y\) and \(Z\).

\section*{Notice}

In case of conflicts within the programmed axis exchange sequence:
Redundant axis names \(\rightarrow\) ERROR, program abort
In case of conflicts with existing axes in the NC channel:
Identical axis names, different axis numbers \(\rightarrow\) ERROR, program abort
If the logic switch NAM is set, the conflict is cleared as follows:
The axis receives the default axis name from its axis parameter list P-AXIS-00297. The user must ensure a clear definition of the default axis name in the list.

\footnotetext{
<Achsnummer>
The physical assignment of axes takes place via the logical axis number. Mathematical expressions are permissible. The logical axis number must be known in the axis management.
}

\section*{Notice}

In case of conflicts within the programmed axis exchange sequence:
Redundant axis numbers \(\rightarrow\) ERROR, program abort In case of conflicts with existing axes in the NC channel:
Axis number already exists in NC channel \(\rightarrow\) WARNING If the logic switch NBR is set, the conflict is cleared as follows:
The axis request is ignored, i.e. it is not executed.

The axis index defines the location of the axis inside the axis group of the NC channel. It then defines the main and the tracking axes (see the table below). Mathematical expressions are permissible if their results are within the range [0... maximum axis number -1]. The axis index may not yet be assigned an axis.
\begin{tabular}{|l|l|}
\hline Index & Axis configuration \\
\hline \(\mathbf{0}\) & 1st main axis in the machining plane. \\
\hline \(\mathbf{1}\) & 2nd main axis in the machining plane. \\
\hline \(\mathbf{2}\) & 3rd main axis generally perpendicular to the machining plane. \\
\hline \(\mathbf{3}\) & 1. Tracking axis. \\
\hline\(\ldots \mathbf{n}\) & (n-2). Tracking axis. \\
\hline
\end{tabular}

\section*{Attention}

To make programming easier, the axis index input can be left empty for tacking axes. In this case, the next free axis index after index 3 is then assigned automatically to this tracking axis. However, it must be noted that the index of a tracking axis is important with regard to various functionalities. For example, all transformational axes must be arranged after the main axes with no gaps in the case of kinematic transformation (RTCP). In such cases, it is then necessary to program the axis index explicitly for the tracking axes.

\section*{Notice}

In case of conflicts within the programmed axis exchange sequence:
Redundant axis indices \(\rightarrow\) ERROR, program abort
In case of conflicts with existing axes in the NC channel:
Axis index is already assigned in the NC channel, different axis \(\rightarrow\) ERROR, program abort.
With set logic switch IDX is set, the conflict is cleared as follows:
The next free index in the axis configuration of the NC channel is automatically determined for the axis.

Offsets are kept axis-specific. This refers to the following offsets:
- Reference point offset
- Clamping offset
- Tool offset
- Zero offset
- Measuring offset
- Manual mode offset
- Position preset

The keywords in the table below can control the adoption of offsets for the an axis request.
\begin{tabular}{|l|l|}
\hline Keyword & Replacing axes and no adoption of offsets (default) \\
\hline ALL & Adopting all offsets * \\
\hline BPV & Adopting reference point offset \\
\hline PZV & Adopting clamping offset \\
\hline WZV & Adopting tool offset * \\
\hline NPV & Adopting zero offset \\
\hline MOFFS & Adopting measuring offset \\
\hline SOFFS & Adopting manual mode offset \\
\hline PSET &
\end{tabular}

\section*{Attention}
* When the tool is selected, please note the following when adopting tool offsets with \#AX DEF:
- If axes are only swapped (internal axis exchange) by \#SET AX and otherwise no additional axes are specified or requested, all offsets (including tool offsets) are also replaced and continue to remain active. Specifying keywords to adopt offsets has no effect.
If a new tool is then selected, the offsets also exchanged are replaced by the offsets of the new tool.
- As soon as an axis release or an axis request is triggered by \#AX DEF (external axis exchange), the tool offsets are re-adopted in the calculation of the sequence of axes indexed in the tool data. Therefore, any adopted tool offsets are replaced by the current tool offsets. If the original tool offsets in the corresponding axes should continue to apply, a new tool must be selected with offsets which have been adapted to the new axis arrangement.

You are therefore advised to run \#AX DEF when the tool is deselected and to ensure the correct assignment of tool offsets in the data record of a newly selected tool by appropriate parameterisation.

\section*{Example}
\begin{tabular}{|l|l|l|l|l|}
\hline Index of tool offsets in tool data & {\([0]\)} & {\([1]\)} & {\([2]\)} & {\([3]\)} \\
\hline Parameterised tool offsets e.g. for T1 & 50 & 0 & 70 & 20 \\
\hline Axis configuration at program start & X & Y & \(\mathbf{Z}\) & --- \\
\hline Included tool offsets after T1 selection & 50 & 0 & 70 & --- \\
\hline "Internal" \#AX DEF \{Z, X, Y\}: & \(\mathbf{Z}\) & \(\mathbf{X}\) & \(\mathbf{Y}\) & --- \\
\hline Tool offsets are also swapped or & 70 & 50 & 0 & --- \\
\hline "External" \#AX DEF \{Z, X, Y, B\}: & \(\mathbf{Z}\) & \(\mathbf{X}\) & \(\mathbf{Y}\) & \(\mathbf{B}\) \\
\hline Tool offsets are again recalculated depending on T1 & 50 & 0 & 70 & 20 \\
\hline
\end{tabular}

\subsection*{12.2.2.4 Load the default axis configuration (\#AX DEF DEFAULT)}

If DEFAULT is specified, the default configuration contained in the channel parameter list [1] [ \(>819]-5\) can be restored. A combination with logic switches and additional axes requests is also permitted.
```

\#AX DEF DEFAULT
(non-modal)
or
\#AX DEF DEFAULT [NAM, NBR, IDX] { [<Achstauschsequenz> {,<Optionen>} ]} (non-modal)

```

\section*{Example}
\(</>\) Assigning axis names, logical axis numbers and axis indices at program start:

The starting position of the following NC program is the default axis configuration listed in the table.
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}
```

; Set axis configuration
;X axis remains in its place,
;Y axis is released since this axis is not contained
; in the new definition
;Z axis is re-sorted acc. to Index 4,
;Y1 and Z1 axis are requested
%ACHSTAUSCH (axis exchange)
N10 \#AX DEF [X,1,0][Y1,4,2][Z1,5,3][Z, 3, 4]
:

```

Assign axis names, logical axis numbers and axis indices after N10:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline Y1 & & 1 \\
\hline\(Z 1\) & 4 & 2 \\
\hline\(Z\) & 3 & 3 \\
\hline
\end{tabular}

Restoring default axis configuration:
```

%ACHSTAUSCH (axis exchange)
N10 \#AX DEF DEFAULT
:

```

Assigning axis names, logical axis numbers and axis indices after N10:
\begin{tabular}{|c|c|c|}
\hline Axis designation & Logical axis number & Axis index \\
\hline\(X\) & 1 & 0 \\
\hline\(Y\) & 2 & 1 \\
\hline\(Z\) & 3 & 2 \\
\hline
\end{tabular}

\section*{12.3 \\ Dwell time}

\section*{\#TIME <expr>}
(non-modal)

See Section Dwell time (G04), (\#TIME) [〉 83].

\subsection*{12.4 Flushing NC channel (\#FLUSH, \#FLUSH CONTINUE, \#FLUSH WAIT)}

The single NC blocks of an NC program are read in continuously by an interpreter module, converted to an internal representation and output to the NC channel for further processing (program decoding) In the NC channel, the data is then read in by further modules (tool radius compensation, contour preparation, etc.), edited and passed through to the interpolator.

This processing step has a buffering effect on the NC channel with the result that program decoding is always ahead of actual execution of commands in the interpolator.
The buffering effect on the NC channel can then lead to a delay in motion block supply to the interpolator and stop axis motions.

The NC command \#FLUSH and \#FLUSH CONTINUE cancel the buffering effect of the NC channel. All buffered NC blocks are immediately processed and executed in their programmed sequence.

These commands are especially helpful to bridge the time gap of large non-motion relevant NC program parts between two motion blocks and to avoid needless stops in axis motion.

\section*{Attention}

The commands to flush the channel (\#FLUSH, \#FLUSH CONTINUE, \#FLUSH WAIT) can not be used with active TRC, polynomial contouring or active HSC mode.

\section*{Release Note}

As of Build V2.10.1503.08, the commands for flushing the NC channel can be used with active polynomial contouring. In this case, contouring is suppressed in the motion block before and after a command to flush the NC channel.

\section*{\#FLUSH}

The command \#FLUSH forces all NC blocks (control commands, path motions) output to the NC channel to be forwarded immediately to the interpolator. However, decoding of the part program is continued without interruption.

At the end of the last path motion before \#FLUSH the interpolator normally stops for a short time even if the next motion block is already presented by the program decoding which is still running.

\section*{Programing Example}

\section*{Flush NC channel}

A WHILE loop containing many non-motion relevant NC blocks is programmed between the motion blocks N10 and N210 (e.g. parameter calculations, variable accesses etc.). As a result, the motion block N10 in NC channel and the preparation of motion block N210 are delayed. The parameter \#FLUSH forces the NC channel to pass on all motion blocks to the interpolator before the WHILE loop. It can then start with executing the path motion up to and including N10.
```

N05 G01 F1000 G90 X150
N10 X200
N20 \#FLUSH
N30 \$WHILE
N40 P1 = [P2 * P3] + V.E...
N50 V.L....
N200 \$ENDWHILE
N210 X250
M30

```


Fig. 95: Mode of operation of \#FLUSH between 2 motion blocks

\section*{\#FLUSH CONTINUE}

A stop at block end of the last motion block can be avoided by adding CONTINUE. If the next motion block is already present in the interpolator, the motion is executed without any interruption. Only if no further motion block is present does the interpolator go to wait state.

Programing Example

A WHILE loop containing many non-motion relevant NC blocks is programmed between the motion blocks N10 and N210 (e.g. parameter calculations, variable accesses etc.). As a result, the motion block N10 in NC channel and the preparation of motion block N210 are delayed. The parameter \#FLUSH CONTINUE forces the NC channel to pass on all motion blocks to the interpolator before the WHILE loop. It can then start with executing the path motion up to and including N10. If the next motion block N210 is present in the interpolator before the end of N 10 , the motion is continued without interruption.
```

N05 G01 F1000 G90 X150
N10 X200
N20 \#FLUSH CONTINUE
N30 \$WHILE ...
N40 P1 = [P2 * P3] + V.E...
N50 V.L....
N200 \$ENDWHILE
N210 X250
M30

```


Fig. 96: Mode of operation of \#FLUSH CONTINUE between 2 motion blocks
```

\#FLUSH WAIT
(non-modal)

```

As opposed to \#FLUSH or \#FLUSH CONTINUE the command \#FLUSH WAIT also interrupts program decoding. All NC blocks before this command are output to the NC channel and executed in interpolator The NC channel is then "flushed". All axes are stopped and the interpolator and program decoder have the same state, i.e. they are synchronised. Only when this state is reached does program decoding continue automatically.

\section*{Attention}

If synchronised or unsynchronised (MOS) M/H functions or \#MSG commands were previously output to the PLC, the controller waits until they are adopted and synchronised by the PLC. Program decoding is interrupted until this state is reached.

\section*{Programing Example}
\#Flush Wait

The parameter \#FLUSH WAIT interrupts program decoding and forces the NC channel to pass on all motion blocks to the interpolator before the WHILE loop. When the interpolator has executed block N10 and position X200 is reached, program decoding continues.
```

N05 G01 F1000 G90 X150
N10 X200
N20 \#FLUSH WAIT
N30 \$WHILE ...
N40 P1 = [P2 * P3] + V.E...
N50 V.L....
N200 \$ENDWHILE
N210 X250

```
M30


Fig. 97: Mode of operation of \#FLUSH WAIT between 2 motion blocks

\subsection*{12.5 Cross-block comments (\#COMMENT BEGIN/END)}
```

\#COMMENT BEGIN Start of comment block Disabled area.
All lines are ignored by the NC kernel without evaluation.
\#COMMENT END End of comment block

```

Both commands must be programmed in a separate NC line, i.e. further NC commands in the same block are not permissible. An exception is a line-by-line comment that must be bracketed "(" and ")".

Comment blocks can be defined inside or outside an NC program. However, they are restricted to the current file since calls from and returns from global subroutines included in the comments are not evaluated.

Application-specific nesting of comments can be enabled.

\section*{Programing Example}

\section*{Cross-block comments}
```

:
\#COMMENT BEGIN Start of comment block 1
\#COMMENT BEGIN Start of comment block 2
\#COMMENT END End of comment block 2
\#COMMENT END End of comment block 1
:
Nesting disabled:
:
\#COMMENT BEGIN Start of comment block 1
\#COMMENT BEGIN Start of comment block 2
\#COMMENT END End of comment block 1 and 2

```

In the following cases, the NC kernel generates error messages:
- "End of File" is read in within a comment block.
- \#COMMENT END is read in without a previously programmed \#COMMENT BEGIN.
- After commend commands, further NC commands are programmed in the same block.

\subsection*{12.6 Waiting for event (\#WAIT FOR)}

This command stops the decoding of the NC program until the arithmetical expression is fulfilled (TRUE or >0.5).

\section*{Programing Example}

\section*{Waiting for event}

N10 \#WAIT FOR V.E.EXT1 == 5 (Decoding of NC program is stopped at this)
(point until the value of the external variable)
(is 5.)

N100 \#WAIT FOR V.E.EXT2
(Decoding of the NC program is stopped at this
(Point until the value of the
(external variable \(>\) is 0.5.)

\section*{Attention}

The controller look-ahead mechanisms may result is the state that a certain number of motion-relevant NC blocks programmed before \#WAIT FOR are retained in the NC channel. If the retained motion blocks are relevant for generating the event itself, a deadlock is caused and execution of the NC program is stopped without any apparent reason.
By programming \#FLUSH or \#FLUSH CONTINUE directly before \#WAIT FOR, the execution of all preceding motion blocks is forced and a deadlock is avoided.

\section*{Programing Example}

Waiting for event
```

Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx \#FLUSH CONTINUE ;Forced execution of all buffered motion
blocks
;Forced execution of all buffered motion
blocks

```

\subsection*{12.7 Adapting minimum radius for tangential feed ((\#TANGFEED))}

\section*{Release Note}

As of Build V2.11.2010.02 the command \#TANGFEED [...] replaces the command \#SET TANGFEED RMIN [...]. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

This command supplements the tangential adaptation of feed programmed with G10/G11 [ 559]. The minimum radius is considered in circular blocks containing the identifier for feed adaptation. The feed is only corrected if the programmed circular radius is greater than or equal to the minimum radius.
```

\#TANGFEED [ RMIN<expr> ] (modal)

```

Minimum contour radius Rmin in [mm, inch] up to which the tangential feedrate is corrected.


Fig. 98: Programming the tangential feedrate.
- A new programmed minimum radius is taken into account with the next relevant circular motion block.
- It is optional to specify the minimum radius. Tangential feed adaptation with G 11 [ 559] is also executed if \#TANGFEED [RMIN...] is not programmed or the radius is assigned the value zero.

\section*{Programing Example}

\section*{Adapting minimum radius for tangential feed}
```

(Contour with TRC and tangential feed adaptation)
N10 V.G.WZR=5
N20 \#TANGFEED [RMIN=3] (Define minimum radius rmin = 3mm)
N30 G41 G01 X0 Y20 G11 F600 (Select TRC and feed adaptation)
N40 X20 Y20
N50 G03 X40 R10 (Feed adaptation)
N60 G01 Y40
N70 G02 X44 R2 (No feed adaptation -> rprg < rmin)
N80 G01 Y20
N20 \#TANGFEED [RMIN=6] (Define new minimum radius rmin = 6mm)
N90 G03 X54 R5
(No feed adaptation }->\mathrm{ rprg < rmin)
N100 G01 Y50
G40 Y20 G11 G10 (Deselect TRC and feed adaptation)
N120 X0 Y0

```

\subsection*{12.8 Suppressing offsets (\#SUPPRESS OFFSETS)}

In combination with a motion block, this command causes an execution of the programmed axis positions without consideration of the active offsets.

All offsets in the NC block are suppressed if a particular offset type is not specified.

\section*{Release Note}

As of Build V2.11.2032.07 it is also possible to select which active offsets (zero points, PSET, measurements etc.) are to be suppressed in the NC block.

Offsets due to active kinematic and/or Cartesian transformations (CS, ACS, ROTATION) are not suppressed by the command. To suppress all kinds of offsets, the command \#MCS ON/OFF [ 743] must be used as an alternative.
```

\#SUPPRESS OFFSETS [ [ ZERO ADD_ZERO PSET CLAMP TOOL MEASURE MANUAL ] ]
<axis_name><expr> {<axis_name><expr>} (non-modal)

```

ZERO
ADD_ZERO
PSET
CLAMP
TOOL
MEASURE
MANUAL
<axis_name><expr>

Zero offsets
Additive zero offsets and reference point offsets
Position presets
Clamping offsets
Tool offsets
Measuring offsets
Manual mode offsets
Axis positions which are moved without offsets.

\section*{Programing Example}

\section*{Suppressing offsets}
```

%suppress_offsets
;Define zero offsets for G54
N05 V.G.NP[1].V.X=11
N10 V.G.NP[1].V.Y=22
;Select zero offsets X11 Y22
N15 G54
;Select and define additive zero offsets X10 Y20
N20 G92 X10 Y20
N25 X100 Y150 ;Position X=121 Y=192
;Suppress all active offsets
N30 \#SUPPRESS OFFSETS X50 Y100 ;Position X=50 Y=100
;Suppress additive offsets
N35 \#SUPPRESS OFFSETS [ADD_ZERO] X50 Y100 ;Position X=61 Y=122
N40 X200 Y250 ;Position X=221 Y=292
N99 M30

```

\subsection*{12.9 Settings for measurement}

\subsection*{12.9.1 Switching measurement type (\#MEAS MODE)}

Using the following command, the measurement type can be defined by the NC program.

\section*{\#MEAS MODE [ [<expr>]}
(modal)
<expr>
Measurement type as described in Section Switching the measurement type [> 90].

Programming \#MEAS MODE without parameters selects the default measurement type which is specified in the channel parameter P-CHAN-00057.

\section*{Programing Example}

\section*{Switch measurement type}
```

N10 \#MEAS MODE [3]
N20 G100 X150
N100 \#MEAS MODE

```
(Measurement type 3)
(Measurement motion block)
(Default measurement type)
```

M30

```

\subsection*{12.9.2 Extended programming (\#MEAS, \#MEAS DEFAULT)}

Alternatively to the \#MEAS MODE command, the following command offers additional measuring parameters. The selected parameter settings remain valid until program end. When the program is restarted, the default settings in the configuration lists are again valid. To be able to change the measurement parameters of an axis, it must be identified as a measurement axis (i.e. the axis parameter P-AXIS-00118 must be set to 1).
```

\#MEAS [ [TYPE<expr>] [ERR_NO_SIGNAL<expr>]
[[SIMU_OFFSET<expr>]|[TRIGGER]]|
[ {AX<Achsname> | AXNR<expr>} [SIGNAL<ident>] [EDGE<ident>] [INPUT<expr>]
[G107|G108]]] (modal)

```

TYPE<expr>

ERR_NO_SIGNAL<expr>

SIMU_OFFSET<expr>

TRIGGER

New measurement type as described in Section Measuring functions [ 90]. This measurement type is valid until the next change or until program end.
Reaction to undetected probe signal:
0 : no error message
1: Error message when measurement run is deselected (default)
This keyword is only effective in particular for a measuring simulation in connection with the axis parameter P-AXIS-00112=4. The value in [mm, inch] offsets the simulated default measuring point relative to the path motion based on the programmed target points.
For measurement type 2 the default measuring point can be shifted in positive or negative direction by SIMU_OFFSET. Any additional offset by the axis parameter [2] [ 819]P-AXIS-00114 is not considered here.
Triggering a programmed probe signal. Only used in conjunction with the global block edge banding function G107/G108 [ 107]. Is only effective if P-CHAN-00257 is active. Notice

SIMU_OFFSET and TRIGGER are exclusive and may not be programmed in combination with the other axisspecific keywords.

AX<Achsname> Name of the axis whose measurement parameters are to be changed. The axis must be configured as a measurement axis.
AXNR<expr> Logical number of the axis whose measurement parameters are to be changed. The axis must be configured as a measurement axis, positive integer
SIGNAL<ident> Name of the probe signal source used for the measurement (see P-AXIS-00516). Valid identifiers:

PLC: Probe signal via PLC
DRIVE:
FIXED_STOP:
PLC_FIRST_EVENT:
PLC_EXT_LATCH_CONTROL:

Probe signal via drive position latch
Probe signal by moving to fixed stop
Probe signal by PLC. Measurement run terminates when one axis receives the measuring event
Measure with measuring interval for external hardware (see [HLI//Measuring with measuring interval for external hardware])
\begin{tabular}{|c|c|}
\hline EDGE<ident> & Relevant measurement edge, see also P-AXIS-00518.. Valid identifiers: \\
\hline & POS: Positive (rising) measurement edge \\
\hline & NEG: Negative (falling) measurement edge \\
\hline INPUT<expr> & Name of the drive measurement input used for the measurement (see P-AXIS-00517). \\
\hline & For probe signal DRIVE: \\
\hline & 1: 1st measurement input \\
\hline & 2: 2nd measurement input \\
\hline & For probe signal PLC_EXT_LATCH_CONTROL: \\
\hline & 1 .. 255: Channel number of measurement input of the external measuring hardware \\
\hline G107 & Deselect edge banding function for this axis, i.e. no measured value is latched for this axis with edge banding. \\
\hline G108 & Select edge banding function for this axis. The precondition is that the "edge banding function must be active for the axis in the axis parameters (see P-AXIS-00098). \\
\hline
\end{tabular}

The \#MEAS command provides the following extensions for a modal measurement run (over several motion blocks) in conjunction with the edge banding function (measurement type 8):
\begin{tabular}{lll}
\hline \#MEAS [ON | OFF] [ [<Messparameter>] ] & (modal) \\
ON & \begin{tabular}{l} 
Select measurement run for measurement type 8 All set or programmed axes are then \\
measured in the following motion blocks.
\end{tabular} \\
OFF & Deselect measurement run for measurement type 8
\end{tabular}

\section*{\#MEAS DEFAULT [ [ AX < Achsname> | AXNR<expr>\}] (modal)}

DEFAULT Reset the axis-referred parameter settings changed by the MEAS command (SIGNAL, EDGE, INPUT, G107/G108) The measurement settings of the axis parameter list are effective again.

\section*{Notice}

The measurement edge (EDGE) and measurement input (INPUT) cannot be changed for SERCOS drives with position latch in the drive (SIGNAL=DRIVE) because this also requires parameter changes in the drive for this purpose.

Programing Example
Setting measurement parameters:

Selecting a different measurement type:
N100 \#MEAS [TYPE=2]

Setting measurement positions for the measurement simulation for measurement type 2 :


For all other measurement types only an offset in negative direction (opposite to the path motion direction) is possible.


Activate measurement by moving to fixed stop for X and Y axes:
N100 \#MEAS [AX=X AX=Y SIGNAL=FIXED_STOP]

Activate probe signal via PLC to negative edge:
N100 \#MEAS [AXNR=1 SIGNAL=PLC EDGE=NEG]

Deactivate edge banding function for Y and Z axes:
N100 \#MEAS [AX=Y AX=Z G107]

Restore the measurement settings in the axis parameters for all path axes:
N100 \#MEAS DEFAULT

Restore the measurement settings in the axis parameters for X axis:
N100 \#MEAS DEFAULT [AX=X]

Modal measurement run with all measurement axes:
```

N5 \#MEAS [TYPE=8]
N10 G01 X100 Y100 F1000
N20 G01 Z200
N30 G01 X200 Y200
N40 \#MEAS OFF

```

Modal measurement run with measurement in X and Y axes, probe signal via PLC, leading edge, no error message if probe signal not detected

N5 \#MEAS ON [TYPE=8 AX=X AX=Z SIGNAL=PLC EDGE=POS ERR_NO_SIGNAL=0]
N10 G01 X100 Y100 Z10 F1000
N20 G01 X200 Y150 Z25
N30 \#MEAS OFF

\subsection*{12.10 Selecting position preset (\#PSET)}

A new position preset is selected by:
```

\#PSET <axis_name><expr> {<axis_name><expr>}
(non-modal)

```
<axis_name><expr> New actual position of the axis in [mm, inch]

The actual positions are programmed as absolute. No path motion is linked to this command.
Axes which have no new position preset programmed retain their current position preset, i.e the corresponding offset does not change for the axis concerned.

The \#PSET command can be repeated as often as required.
The offset of the coordinate system caused by the \#PSET command is not valid across programs.
The offset of the coordinate system caused by the \#PSET command deselected by the homing function (G74).

\subsection*{12.10.1 Deselecting position preset (\#PRESET)}
\#PRESET \{<axis_name> <dummy_expr>\}
(non-modal)
<axis_name><dummy_expr>
The axis position preset is reset. The coordinate value in [mm, inch] is only required for syntax reasons, otherwise it is irrelevant.

If \#PRESET is programmed without specifying an axis, the position preset is deselected in all axes.

\section*{Notice}

If tool radius compensation, mirroring or diameter programming are selected, \#PSET or \#PRESET may not be programmed.
("non modal" only applies to the commands \#PSET and \#PRESET. The position preset offset itself remains valid of course until it is again selected or deselected by \#PRESET.)

\section*{Programing Example}

Deselect position preset
```

N10 X50 Y10 Z0
N20 \#PSET X20 Y20 (Set position for X and Y)
N30 X10
N40 \#PRESET X10
N50 X30
N60 \#PRESET
N70 M30

```

The figure below shows the position of the x axis in machine coordinates after execution of a particular NC block:


Fig. 99: Positions of the \(X\) axis in machine coordinates / programmed coordinates. (In this example, no other coordinate transformations are selected).

\subsection*{12.11 Synchronous operation}

\subsection*{12.11.1 Programming axis couplings (\#SET AX LINK, \#AX LINK)}

The following NC command can be defined in an NC program for axis couplings:
```

\#SET AX LINK [ <coupling_group>, <Slave>=<Master> {,<Slave>=<Master>} ]

```
or alternatively
\#AX LINK [NBR] [ <coupling_group>, <Slave>=<Master> \{,<Slave>=<Master>\} ]
\begin{tabular}{ll} 
<coupling_group> & Number of the coupling group \({ }^{(1)}\) \\
<Slave> & Designation or logical axis number of the slave axis of coupling pair \(\mathrm{i}^{(2)}\) \\
<Master> & Designation or logical axis number of the master axis of coupling pair \(\mathrm{i}^{(2)}\) \\
NBR & \begin{tabular}{l} 
The logic switch NBR switches over evaluation from axis names to logical axis names. \\
\\
The axis couplings must then be defined with logical axis numbers. The axes need not \\
to be present in the NC channel. Their presence in the NC channel is only checked \\
when the coupling group is activated.
\end{tabular}
\end{tabular}

\section*{Notice}

At least one master-slave coupling pair must be defined for each coupling group.
The coupling of spindles is described in greater detail in Section Synchronous spindle operation [ \({ }^{\text {b }}\) 687].

\section*{General handling and method of operation:}
- The definitions of coupling groups apply program global. This means that they can be selected in a subsequent NC program (see also Section 11.11.3 [> 357]).
- The coupling group with coupling number ' 0 ' cannot be defined in the NC program. It is defined in channel parameter [1] [> 819]-2.
- A disabled coupling group can be modified at any time. The previously defined couplings are overwritten.
- An enabled coupling group may not be modified.
- Recursive axis couplings are not permissible. A master axis in a coupling group cannot be a slave axis at the same time and vice versa. A slave axis in one coupling group cannot be a master axis in another coupling group at the same time and vice versa.
- Master and slave axis of a coupling pair may not be identical.
- A main axis may not be a slave axis.
- A slave axis can only be assigned to one master axis, but a master axis can have several slave axes.
- When synchronous mode is activated, a check is made whether all master and slave axes exist in the axis group of the NC channel.
- Master and slave axes must be of the same axis type and be used in the same axis mode.
- The NC command used for programming the coupling group must be a single instruction in the NC block.
- The number of the coupling group may also be programmed with mathematical expressions. The result must be a positive integer.
- A slave axis may not be an active tracking axis at the same time (\#CAXTRACK).

1 ... [Max. number of coupling groups-1], see [6] [> 819] -2.11
Max. number of coupling pairs, see [6] [ 819] -2.12

\section*{Programing Example}

Programming axis couplings
```

N10 \#SET AX LINK[1, Z2=Z]
N20 \#SET AX LINK[2, Y2=Y,
X2=X]
N30 \#SET AX LINK[3, X1=X,
X2=X, X3=X]
or alternatively
N10 \#AX LINK[1, Z2=Z]

```
N20 \#AX LINK NBR Coupling via log. axis numbers

\subsection*{12.11.2 Extended programming of axis couplings ("SOFT-GANTRY") (\#SET AX LINK, \#AX LINK)}

\section*{Release Note}

The availability of this function depends on the configuration and the scope of the version.

Path axes can also be used as so-called gantry axes. Contrary to normal synchronous mode, additional position deviation monitoring mechanisms are active and specific error reactions apply. Due to the machine structure, the axes are permanently linked to each other in mechanical (and static) gantry operation and defined by the machine configuration. After controller start-up it not possible to make a dynamic change in the gantry coupling (see figure below).

In addition to path axes, spindles can also be run in synchronous mode. A detailed description is contained in Section Spindle synchronous mode [ 687].


Fig. 100: Mechanical gantry operation
Machines that do not permit any mechanical gantry operation because of their basic structure, e.g. milling machines with two independent slides, can be operated in gantry mode by programming. For example, this is necessary when slides must be linked to one another for clamping and machining large workpieces (see Figure below)


Fig. 101: Programmable gantry operation ("soft" gantry)
The \#SET AX LINK and \#AX LINK commands described in Section Programming axis couplings [ \(>355\) ] are available in an extended syntax for this purpose:
```

\#SET AX LINK [ <coupling_group>, [ <Slave>=<Master>,G [,<limit_1>, limit_2>]]
{, [ <Slave>=<Master>,G [,<limit_1>, limit_2>]]}]
or alternative
\#AX LINK [NBR] [ <coupling_group>, [ <Slave>=<Master>,G [,<limit_1>, limit_2>] ]
{, [ <Slave>=<Master>,G [,<limit_1>, limit_2>]] }]

```
\begin{tabular}{ll} 
<coupling_group> & Number of the coupling group \({ }^{(1)}\) \\
<Slave> & Designation or logical axis number of the slave axis of coupling pair \(\mathrm{i}^{(2)}\) \\
<Master> & Designation or logical axis number of the master axis of coupling pair \(\mathrm{i}^{(2)}\) \\
NBR & \begin{tabular}{l} 
With the logic switch NBR the evaluation can be changed from logical axes names to \\
axes numbers. The axis couplings must then be defined with logical axis numbers. The \\
axes need not be present in NC channel. Their availability in NC channel is checked \\
only at activation of the coupling group! \\
Keyword for "gantry coupling"
\end{tabular} \\
G &
\end{tabular}
(1) 1 ... [Max. number of coupling groups-1], see [6] [> 819]-2.11
(2) Max. number of coupling pairs, see [6] [ 819] -2.12

When gantry coupling is used, the following values serve to monitor in two stages the permissible position difference of the gantry axes. Specified in [mm]. Positive real number:
<limit_1>
<limit_2>
First monitoring limit in [mm, inch]. If this limit is exceeded, the motion is aborted and the controller assumes the error state. In the case of a default, the position difference is eliminated when RESET is executed. Depending on the specific application, a deviating motion can also be implemented
Second monitoring limit in [mm, inch]. An error that cannot be RESET is output if this limit is exceeded. The controller must be switched off and the position difference must be eliminated manually.

\section*{Notice}

If the monitoring limits are not programmed, the defaults apply from the axis parameter data records P-AXIS-00072 and P-AXIS-00071 of the slave axis.

\section*{Additions to general handling and mode of operation}
- Gantry coupling takes place precisely at the positions where the axes are located at the time (see Section Extended programming of axis couplings (SOFT-GANTRY) [ 357]) when the coupling is selected. There is no need to specify an offset in the NC command because the offset is calculated internally in the position controller via the command positions.
- The dynamic data of the slave axis is taken into account in the contouring motion.
- For safety reasons, a coupling which is still active on RESET or at program end is implicitly restored the next time the program is started. This reaction is parameterisable (P-CHAN-00104, P-CHAN-00105).

\section*{Programing Example}

Extended programming of axis couplings (SOFT-GANTRY)
```

N10 \#SET AX LINK[1, [Y2 =
Y1,G,0.01,0.25]]
N20 \#SET AX LINK[2, [Y2 = Y1,G]]
N30 \#SET AX LINK [3,[Y2 = Y1]] Standard coupling of Y2 with Y1. No gantry oper-
ation.
or alternative
N10 \#AX LINK[1, [Y2 = Y1,G,0.01,0.25]]
N20 \#AX LINK NBR[2, [8 = 2,G]] Gantry coupling via logical axis numbers

```

The parallel machining of workpieces with a symmetrical or scaled contour can also be programmed by means of an extended syntax of the \#SET AX LINK and \#AX LINK commands. Position differences are not monitored in these modes (mirroring or scaling).
```

\#SET AX LINK [ <coupling_group>, [ <Slave>=<Master>,<nominator>, <denominator> ]
{, [ <Slave>=<Master>,<nominator>, <denominator> ] } ]

```
or alternative
\#AX LINK [NBR] [ <coupling_group>, [<Slave>=<Master>, <nominator>, <denominator>]
    \{, [<Slave>=<Master>,<nominator>, <denominator>] \}]
\begin{tabular}{ll} 
<coupling_group> & Number of the coupling group \({ }^{(1)}\) \\
<Slave> & Designation or logical axis number of the slave axis of coupling pair i \({ }^{(2)}\) \\
<Master> & Designation or logical axis number of the master axis of coupling pair \(\mathrm{i}^{(2)}\) \\
NBR & \begin{tabular}{l} 
With the logic switch NBR the evaluation can be changed from logical axes names to \\
axes numbers. The axis couplings must then be defined with logical axis numbers. The \\
axes need not be present in NC channel. Their availability in NC channel is checked \\
only at activation of the coupling group!
\end{tabular} \\
\begin{tabular}{ll} 
<numerator>, <denominat- \\
or> & Integers. Their purpose is to calculate a coupling factor between the master and slave \\
& axes. \\
& The following applies to the resulting coupling factor: \\
& -1 : mirror coupling \\
\(1:\) standard coupling; equivalent to the previous syntax \\
\(0:\) output of an error message
\end{tabular}
\end{tabular}

\section*{Attention}

Factors that result in pure scaling (factor \(\neq 1\) ) or in scaling with simultaneous mirroring (factor \(\neq-1\) ) are currently not permitted. A warning is output and the coupling is handled as a standard coupling. In other word, only the coupling factors \(\mathbf{1}\) and \(\mathbf{- 1}\) are permissible (see examples).n
(1) 1 ... [Max. number of coupling groups-1], see [6] [> 819]-2.11
(2) Max. number of coupling pairs, see [6] [> 819] -2.12

\section*{Programing Example}
```

N10 \#SET AX LINK[1, [Y2 = Y1,1,-1]]
N20 \#SET AX LINK[1, [Y2 = Y1,-1,1]]
N30 \#SET AX LINK[1, [Y2 = Y1,-2,2]]
N40 \#SET AX LINK[1, [Y2 = Y1,1,1]]
N50 \#SET AX LINK[1, [Y2 = Y1,2,2]]
N60 \#SET AX LINK[1, [Y2 = Y1,0,1]]
N70 \#SET AX LINK[1, [Y2 = Y1,1,0]]
N80 \#SET AX LINK[1, [Y2 = Y1,1,2]]
N90 \#SET AX LINK[1, [Y2 = Y1,2,3]]
N100 \#SET AX LINK[1, [Y2 = Y1,3,2]]
N110 \#SET AX LINK[1, [Y2 = Y1,-1,2]]
N120 \#SET AX LINK[1, [Y2 = Y1,-3,2]]
or alternative
N40 \#AX LINK[1, [Y2 = Y1,1,1]]
N50 \#AX LINK NBR[1, [8 = 2,2,2]]

```

\subsection*{12.11.3 Enabling/disabling axis couplings (\#ENABLE AX LINK, \#DISABLE AX LINK)}

A coupling group can be activated by the following NC command:
```

\#ENABLE AX LINK [<coupling_group>]
or
\#ENABLE AX LINK (Coupling group 0, defined in the channel parameter list)
or alternative
\#AX LINK ON [<coupling_group>]
or
\#AX LINK ON (Coupling group 0, defined in the channel parameter list)

```

An active coupling group can be deactivated by the following NC commands:
```

\#DISABLE AX LINK [<coupling_group>]
or
\#DISABLE AX LINK (Deselect the last activated coupling group)
or alternative
\#AX LINK OFF [<coupling_group>]
or
\#AX LINK OFF (Deselect the last activated coupling group)
\#AX LINK OFF ALL (Deselect all active coupling groups)

```

\section*{Handling and method of operation:}
- No coupling group is active after start-up in the initial position of the NC kernel. Activation of axis coupling begins with programming in the NC program and ends, if not cancelled, when the program ends (M30, M02). If active axis couplings are to remain effective for the next program, i.e. program global, a specific channel parameter P-CHAN-00105 must be set.
- Multiple coupling groups can be enabled at the same time.
- Unassigned coupling groups cannot be activated. A coupling group is considered assigned if at least one valid master-slave coupling pair was defined.
- The NC command must be a single instruction in the NC block.
- The number of the coupling group can also be programmed via mathematical expressions.
- WRK must not be selected when synchronous operation is selected or cancelled.
- Manual mode with parallel interpolation (G201) may not be active for the slave axes when synchronous mode is selected
- Positions of slave axes may not be addressed in the NC program when synchronous mode is active.

\section*{Programing Example}

Axis designations used: Master axis system \(\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \mathrm{C}\)
Slave axis system Y_S, Z_S, C_S
```

(Initialisation program)
%L UP_INIT_ACHS_KOPPL
(initialise axis coupling 1)
N10 \#SET AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
(or \#AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
N20 M17
(tool changing program)
%L UP WZ
N30 \#DISABLE AX LINK (or \#AX LINK OFF)
(Approach tool change position)
N40 G01 G90 Y1000 Z100 C0 Y_S=1000 Z_S=100 C_S=0
(Tool change; T10 contains all tool axis offsets and the tool lengths of master and
slave tools; or these values are explicitly included in the calculation.) )
N50 T10 D10
(Further commands for physical tool change)
(Approach old coupling position. The coupling position may also be defined via para-
meter programming, and then be used by the subroutine.)
N80 G01 G90 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50
N90 \#ENABLE AX LINK[1] (or \#AX LINK ON[1])
N110 M17
(Subroutine for contour machining)
%L UP1
N150 G01 G91 X10 Y10 Z-20 C90
N160 G02 X20 Y20 I10 J10
N170 LL UP WZ
N180 G01 G911 X10 Y10 Z-20 C90
N190 G02 X20 Y20 I10 J10
N200 M17

```
(Main program; initial condition: Both tools were changed.)
(Move both axis systems to coupling position first.)
N300 G01 G91 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50 F300
(Start synchronous operation)
N310 \#ENABLE AX LINK[1] (or \#AX LINK ON[1])
N320 LL UP1
..........................................

N400 \#DISABLE AX LINK (or \#AX LINK OFF)
N410 M30

\subsection*{12.11.4 Inquiring coupling state and coupling number via variables}

Variables in the NC program can be used to request the current coupling state and the current or last active coupling group. The following new axis group-related variables are introduced for this purpose:

Inquiring the current or last active coupling group:
V.G.AX_LINK.NR

Inquiring the current coupling state:

\section*{V.G.AX_LINK.ACTIVE}

Only read access is permissible for these variables. A write operation leads to an error message and to program abort.

\subsection*{12.12 Messages from the NC program}

This functionality permits messages from the NC program to be output to any system device. The functionality corresponds to the 'printf' function in the operating systems.

Messages in the NC program can be programmed, e.g. to notify the operator of the current state of the process or the controller. Therefore, the message display may be optionally synchronised with the current processing state. This means that the message is only displayed when the interpolator reaches the point in the NC program.

A message may consist
- of ASCII characters,
- parameters and
- variables.

A message may not be programmed together with other NC commands in the same NC block (exception: \#ADD).

The message is sent in ASCII format.

\subsection*{12.12.1 Programming a message (\#MSG)}
\#MSG [<mode>] [<receiver>] ["<message_text>"]

\section*{<mode>}
\begin{tabular}{|l|l|}
\hline Mode & Time of message output \\
\hline SYN & Synchronous with processing state in interpolator \\
\hline SYN_ACK & \begin{tabular}{l} 
Synchronous with processing state in interpolator with acknowledge- \\
ment by the receiver (e.g. SPS). Processing is only continued in the in- \\
terpolator after the acknowledgement is received.
\end{tabular} \\
\hline--- & Immediately after decoding (default). \\
\hline ACK & \begin{tabular}{l} 
Immediately after decoding with acknowledgement by the receiver (e.g. \\
SPS). Decoding is only continued after the acknowledgement is re- \\
ceived.
\end{tabular} \\
\hline
\end{tabular}
<receiver>
The receiver of the message is specified by its receiver ID. If no receiver is specified, it is sent to the default receiver.
\begin{tabular}{|l|l|}
\hline Receiver ID & Description \\
\hline ISG_DIAG_BED & Message is sent to the CNC diagnosis interface (default) \\
\hline DIAG & Message is sent to diagnosis data, see P-CHAN-00514 \\
\hline HMI & Message is sent to the system-specific user interface \\
\hline PLC & Message is sent to the system-specific PLC \\
\hline
\end{tabular}

\section*{Attention}

Messages must be read by the corresponding receiver, otherwise the CNC stops after sending 10 messages.
<message_text> The message text must be enclosed in quotation marks "...".

\section*{Programing Example}

Programming a message
```

\#MSG HMI ["Text_1"]
\#MSG SYN HMI ["Text 2"]
\#MSG ["Text_3"] (message sent to default receiver)
\#MSG SYN ["Text_4"]
\#MSG SYN_ACK ["Text_5"]

```

Text_1 is sent immediately after decoding to the system-specific user interface; the Text_2 string is sent synchronously for processing in the interpolator.

Text_3 is sent immediately after decoding to the default receiver; the Text_4 string is sent to the same device synchronously for processing in the interpolator.

Text_5 is also sent to the default receiver synchronously for processing in the interpolator but processing is stopped until the acknowledgement is received.

The following format elements to output numerical values and to output strings are available:
\begin{tabular}{|l|l|}
\hline \%d or \%D & Outputting decimal numbers with sign (signed integer) \\
\hline \%u or \%U & Outputting decimal numbers without sign (unsigned integer) \\
\hline \%f or \%F & Outputting real numbers (float) \\
\hline \%s or \%S & Outputting strings (<String>, Macros, V.E.STRING) \\
\hline
\end{tabular}

\section*{Notice}

A maximum of 10 parameters can be output with \%xx. The number of format characters must match the number of subsequent parameters.

\section*{Programing Example}

\section*{Programming a message with format elements:}
```

\#MSG [ "Message text_%d and message text_2", 1 ]

```

Sent string: Message text_1 and message text_2
\#MSG [ "Current measured value: \%f", V.A.MEAS.ACS.VALUE.X]
Sent string: Current measured value: \(3.4567800000 \mathrm{E}+001\)
\#MSG [ "Current state: \%s", "End of roughing"]
Sent string: Current state: End of roughing
The control sequence "\%\%" must be programmed to output the "\%" character. The "l" character must precede quotation marks.

\section*{Programing Example}

Programming a message with "\%" and quotation marks:
```

\#MSG [ "Text with %% character"]
\#MSG [ "Text in quotation marks: \"TEXT\" "]

```

Sent string: Text with \% character and text in quotation marks: "TEXT" A message text may also be specified with parameters and variables.

\section*{Programing Example}

\section*{Programming a message with parameters and variables:}
```

P10 = 1
V.P.BSP = 2
\#MSG SYN ["Text_%D and Text_%D", P10, V.P.BSP]

```

This example sends the Text_1 and Text_2 strings synchronously to processing in the interpolator to the default receiver.

\subsection*{12.12.2 Programming message information \#MSG INFO)}

The following command can be used to program user-specific information and display parameters for message processing. Information and parameters are also sent as an ASCII string.
\#MSG INFO [<mode>] [<receiver>] ["<message_information>"]

Compared to a message in which the message receiver usually displays only the ASCII string, message information indicates that the ASCII string contains user-specific information that is encoded by the receiver itself.

SYN mode must be specified if the message information is to be sent to the receiver synchronously to the processing state in the interpolator. The message is output immediately after decoding if a mode was not specified (see previous section).

The same requirements as for the \#MSG command apply to the receiver specifications (see previous section).

The message information must be enclosed in quotation marks "...". The same format elements are available for the output of numerical values as for the \#MSG command (see previous section).

\section*{Programing Example}
\(\overline{\omega /\rangle} \quad\) Programming message information
```

P1 = 20
\#MSG INFO ["ROT,%d,FETT",P1]

```

In this example, the string RED, \(20, B O L D\) is sent as message information to the default receiver.

\subsection*{12.12.3 Including the 'Macro' functionality}

\section*{Release Note}

The availability of this function depends on the configuration and the scope of the version.

Messages may also be defined as a macro for the message management across NC programs. This is particularly useful for messages that are used repeatedly or by several programs.

Example of a macro content as a message in the channel parameter list [1] [> 819]:
```

makro def[1].symbol Meldung 1

```
makro_def[1].nc_code \#MSG ["Message text"]

This permits the following message to be output in the NC program by the following macro call:
```

"Message_1'
(The "Message text" string is sent to the)
(default receiver of messages)

```
:

\subsection*{12.12.4 Writing messages to a file (\#MSG SAVE)}

The command \#MSG SAVE saves data directly from the NC program to a file on a storage device (e.g. hard disk). The possibilities of structuring a message text and logging data are fully identical to the scope of the commands \#MSG or \#MSG INFO.
\#MSG SAVE [EXCLUSIVE] [CONTINUE] ["<message_text>"]

It is not necessary to specify a mode (SYN) or a receiver ID (e.g. HMI) as this would produce an error message since the message is written directly to the report file in the NC program after evaluation.

The message text must be enclosed in quotation marks "...". Every time \#MSG SAVE is called, the message text is added to the end of the report file that already exists. To create a new report file, the user should delete an existing file before NC program start .
\#MSG SAVE ["message text "] writes the message text to the report file in the format
< time stamp >
"message text".
\#MSG SAVE EXCLUSIVE["message text "] writes the message text to the report file in the format without time stamp in the format
"message text"
\#MSG SAVE CONTINUE ["message text"] can suppress a line break at the end of the message text. The next \#MSG SAVE command then writes in the same line as the previous \#MSG SAVE command.

The name of the report file can be previously defined with the command \#FILE NAME. If this option is not used, \#MSG SAVE writes to file with the name message.txt .

The output file path is defined in the start-up list by P-STUP-00018 using the path type P -STUP-00020. If no path information is entered for the report file, a default path is used depending on the control platform or the report file is stored in the main directory of the NC controller.

\section*{Programing Example}

Writing messages to a file
:
\#FILE NAME [MSG="example.txt"] ;Name of output file
:
\#MSG SAVE["Hello World \%d",12345] ;Write text "Hello World 12345" in
output file example.txt
:

\subsection*{12.12.5 Outputting additional informations at block end (\#ADD)}

The \#ADD command can be used to create additional information in the NC block. The possibilities for structuring this additional information (message texts) are fully identical to the scope of the \#MSG commands. However, as opposed to the \#MSG commands, different NC commands can be programmed before \#ADD. Therefore, \#ADD must always be programmed as the last command at NC block end. The following comments are permitted.

\section*{\#ADD [<receiver>] ["<Zusatzinformationen>"]}

It is not necessary to specify a mode (SYN) as this would produce an error message since the message is automatically always output synchronously to the processing state in the interpolator. The same requirements as for the \#MSG commands apply to the receiver specification.

Additional information must be enclosed in quotation marks "...".

\section*{Programing Example}

\section*{Output of additional informations at block end}
```

%add_block_info
N05 P1=20
N10 G00 X0 Y0 Z0
N15 T1 \#ADD["Werkzeug T=%d aktiv", V.G.T_AKT]
N20 G01 F2000 X10 \#ADD["Approach X position"] (Comment)
N30 YP1 \#ADD["Y-position=%d", P1]
N40 z30 \#ADD["Z-position"]
N50 Z33 \#ADD["Z position"] X11 Y22 ->Error 21509
N999 M30

```

\subsection*{12.13 Jerk limiting slope}

The slope function determines the velocity on the programmed path and maintains the specified permissible velocities, accelerations and jerks [2] [> 819]-1. The following modes are available :
- Step-shaped acceleration profile with restriction of acceleration without monitoring the jerk
- Trapezoidal acceleration profile with jerk monitoring
- Square-sinusoidal acceleration profile with jerk monitoring

The acceleration curve is generated depending on the slope function selected:


Square-sinusoidial acceleration profile
Fig. 102: Acceleration on the programmed path
The acceleration profile is parameterised axis-specific by accelerations and ramp times [2] [ 8 819]-1:


Fig. 103: Parameters of the acceleration profile.

\subsection*{12.13.1 Selecting operating mode (\#SLOPE, \#SLOPE DEFAULT)}

\section*{Release Note}

As of Build V2.11.2010.02 replaces the command \#SLOPE [...] the command \#SET SLOPE PROFIL [...]. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.
```

\#SLOPE [ TYPE<ident> [ NO_OPT<expr> ] ] (modal)

```
TYPE<ident> Type of acceleration profile. Permitted identifiers:

STEP Step-shaped acceleration profile (default)
TRAPEZ Trapezoidal acceleration profile
SIN2 Square-sinusoidal acceleration profile
HSC HSC slope, recommended for "Extended HSC programming" [> 248] *
NO_OPT<expr> Switch the optimised use of jerk:
0 : Optimised use of jerk is active. This reduces processing time but requires greater computing resources. It must be checked whether the existing hardware is adequate.
1: Optimised use of jerk is not active (initial state).

\section*{Notice}
* The use of this feature for selecting the HSC slope profile type requires a licence for the "HSC" extension package. It is not included in the scope of the standard license.

\section*{Notice}

The specific weighting adaptation of ramp time (G132/G133) and acceleration (G130/G131) is no longer supported by the command \#SLOPE [...]. Weightings always act on all ramp times and accelerations (default).

The programming of \#SLOPE DEFAULT restores the initial state (as after start-up). This means that the slope type is set from the channel parameter list P-CHAN-00071.

\section*{Programing Example}
```

N10 G01 X50 Y10 Z0 F1000 (step-shaped accel.profile, default)
N20 \#SLOPE [TYPE=TRAPEZ] (trapezoidal accel.profile)

```
N30 X10 Y30
```

N40 \#SLOPE [TYPE=SIN2] (sinusoidal accel.profile)

```

N50 X15
N60 Y50
N70 M30

The following velocity curve results on the programmed path:


Fig. 104: Velocity curve depending on the programmed path.

\subsection*{12.14 Writing and reading SERCOS parameters and commands}

\subsection*{12.14.1 SERCOS parameters (\#IDENT)}

The following NC commands are used to write and read SERCOS parameters. The original SERCOS format [4] [> 819] is used to simplify programming.

Additional information, e.g. axis name or logical axis number, codes and an attribute, are required to correctly process the parameter IDENT number in the drive. This information is programmed together with the IDENT number in the same NC command.

Notice
"Non-synchronous" means execution of the command in the decoding context.
"Synchronous" means execution of the command synchronously with processing, i.e. at interpolator level.

\subsection*{12.14.1.1 Non-synchronised write (\#IDENT WR)}
```

\#IDENT WR [ AX<axis_name> | AXNR<expr> ID<ldent_nr> VAL<expr> TYPE<expr>
DEC<expr> <Drive_type> ]

```

AX<axis_name>
AXNR<expr>
ID<ldent_nr> VAL<expr>
TYPE<expr>

Name of SERCOS axis
Logical axis number of SERCOS axis, positive integer
ID Number in SERCOS format, e.g. S-0-0047 or P-0-0129
Value to be written; real number
Data type of value (2 or 4 byte length):
\begin{tabular}{|l|l|}
\hline Valid values & Meaning \\
\hline \(\mathbf{2}\) & 2 bytes data length \\
\hline \(\mathbf{4}\) & 4 bytes data length \\
\hline
\end{tabular}

DEC<expr>
<Drive_type>

Number of decimal places; positive integer
Drive type
\begin{tabular}{|l|l|}
\hline Valid IDs & Meaning \\
\hline SERC & SERCOS drive (currently only one permitted) \\
\hline
\end{tabular}

\section*{Programing Example}
```

:
\#IDENT WR [AX X ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
\#IDENT WR [AXNR 1 ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]

```
:

\section*{Attention}

No plausibility check is made for logical axis number, identification number or the programmed attributes of data type and decimal places. The operator is solely responsible for making the correct entries.

\subsection*{12.14.1.2 Non-synchronised read (\#IDENT RD)}
\#IDENT RD [ AX<axis_name> | AXNR<expr> ID<ldent_nr> P =<Variable> TYPE<expr> DEC<expr> <Drive_type> ]
\begin{tabular}{ll} 
AX<axis_name> & Name of SERCOS axis \\
AXNR<expr> & Logical axis number of SERCOS axis, positive integer \\
ID<ldent_nr> & ID Number in SECOS format, e.g. S-0-0047 or P-0-0129 \\
<Variable> & Variable in die der zu lesende Wert abgelegt wird. z.B. P-Parameter oder V.P. , V.L. or \\
TYPE<expr> & V.S. variables. \\
& Data type of value (2 or 4 byte length):
\end{tabular}
\begin{tabular}{|l|l|}
\hline Valid values & Meaning \\
\hline \(\mathbf{2}\) & 2 bytes data length \\
\hline \(\mathbf{4}\) & 4 bytes data length \\
\hline
\end{tabular}

DEC<expr>
Number of decimal places; positive integer
<Drive_type>
Drive type
\begin{tabular}{|l|l|}
\hline Valid IDs & Meaning \\
\hline SERC & SERCOS drive (currently only one permitted) \\
\hline
\end{tabular}

\section*{Programing Example}
```

:
\#IDENT RD [AX X ID S-0-0104 P=P1 TYP 4 DEC 2 SERC] or
\#IDENT RD [AXNR 1 ID S-0-0104 P=V.P.KV_WERT TYP 4 DEC 2 SERC]
:

```

\section*{Attention}

No plausibility check is made for logical axis number, identification number or the programmed attributes of data type and decimal places. The operator is solely responsible for making the correct entries.

\subsection*{12.14.1.3 Synchronised write (\#IDENT WR SYN)}
```

\#IDENT WR SYN [ AX=.. | AXNR=.. ID=.. VAL=.. TYP=..

```
    DEC=.. <Drive_type> [ NO_WAIT ] ]

AX<axis_name>
AXNR<expr>
ID<ldent_nr> VAL<expr>
TYP<expr>

Name of SERCOS axis
Logical axis number of the SERCOS axis, positive integer
ID Number in SECOS format, e.g. S-0-0047 or P-0-0129
Value to be written; real number
Data type of value (2 or 4 byte length):
\begin{tabular}{|l|l|}
\hline Valid values & Meaning \\
\hline \(\mathbf{2}\) & 2 bytes data length \\
\hline \(\mathbf{4}\) & 4 bytes data length \\
\hline
\end{tabular}

DEC<expr>
<Drive_type>

Number of decimal places; positive integer
Drive type
\begin{tabular}{|l|l|}
\hline Valid IDs & Meaning \\
\hline SERC & SERCOS drive (currently only one permitted) \\
\hline
\end{tabular}

\footnotetext{
NO_WAIT
No waiting for successful writing of the parameter.
If this keyword is not programmed, the interpolation always waits for the parameter to be written.
}

\section*{Programing Example}

\section*{Non-synchronised write (IDENT)}
```

:
\#IDENT WR SYN [AX=X ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
\#IDENT WR SYN [AXNR=1 ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
:
\#IDENT WR SYN [AX Y ID S-0-0104 VAL655.35 TYP 4 DEC 2 SERC NO WAIT]

```

\section*{Attention}

No plausibility check is made for logical axis number, identification number or the programmed attributes of data type and decimal places. The operator is solely responsible for making the correct entries.

\subsection*{12.14.2 SERCOS commands (COMMAND)}

The following NC commands serves to start and wait for execution of SERCOS commands. The original SERCOS format [4] [ 819] is used to simplify programming.
Additional information, such as axis name or logical axis number and code, is required for correct processing of the IDENT number command in the drive. This information is programmed together with the IDENT number in the same NC command.

\section*{Notice}
"Non-synchronous" means execution of the command in the decoding context.
"Synchronous" means execution of the command synchronously with processing, i.e. at interpolator level.

\subsection*{12.14.2.1 Non-synchronised write (\#COMMAND WR)}
```

\#COMMAND WR [ AX<axis_name> | AXNR<expr> ID<Ident_nr> <Drive_type> ]

```

\begin{tabular}{|c|c|}
\hline & Programing Example \\
\hline \[
\langle/\rangle
\] & Non-synchronised write (COMMAND) \\
\hline \#COMMAND & [AX=X ID S-0-0148 SERC] \\
\hline \#COMMAND & [AXNR 1, ID S-0-0148, SERC] \\
\hline & Attention \\
\hline - & No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries. \\
\hline
\end{tabular}

\subsection*{12.14.2.2 Synchronised write (\#COMMAND WR SYN)}
\#COMMAND WR SYN [ AX<axis_name> | AXNR<expr> ID<ldent_nr> <Drive_type> ]

AX<axis_name>
AXNR<expr>
ID<Ident_nr>
<Drive_type>

Name of SERCOS axis
Logical axis number of SERCOS axis, positive integer
Identification number of the command in SERCOS format, e.g. S-0-0148 (drive-controlled referencing)
or S-0-0170 (tracing cycle)
Drive type
\begin{tabular}{|l|l|}
\hline Valid IDs & Meaning \\
\hline SERC & \begin{tabular}{l} 
SERCOS drive \\
(currently only one permitted)
\end{tabular} \\
\hline
\end{tabular}

\section*{Programing Example}

Synchronised write (COMMAND)
```

:
\#COMMAND WR SYN [AX Y, ID S-0-0170, SERC]
\#COMMAND WR SYN [AXNR 2 ID S-0-0170 SERC]

```
:

\section*{Attention}

No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.

\subsection*{12.14.2.3 Non-synchronised wait (\#COMMAND WAIT)}
\#COMMAND WAIT [AX<axis_name> | AXNR<expr> ID<ldent_nr> <Drive_type> ]

AX<axis_name> Name of SERCOS axis
\begin{tabular}{|l|l|}
\hline Additional ID & Meaning \\
\hline ALL & Check all SERCOS axes existing in the system \\
\hline
\end{tabular}

AXNR<expr>
ID<Ident_nr>
<Drive_type>

Logical axis number of SERCOS axis, positive integer Identification number of the command in SERCOS format, e.g. S-0-0148 (drive-controlled referencing) or S-0-0170 (tracing cycle) If no identification number is programmed, the system waits for all open commands. Drive type
\begin{tabular}{|l|l|}
\hline Valid IDs & Meaning \\
\hline SERC & \begin{tabular}{l} 
SERCOS drive \\
(currently only one permitted)
\end{tabular} \\
\hline
\end{tabular}

\section*{Programing Example}

Non-synchronised wait (COMMAND)
```

\#COMMAND WAIT [AX X, ID S-0-0148, SERC]
.
\#COMMAND WAIT [AX ALL ID S-0-0148 SERC]

```

\section*{Attention}

No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.

\subsection*{12.14.2.4 Synchronised wait (\#COMMAND WAIT SYN)}
\#COMMAND WAIT SYN [AX<axis_name> | AXNR<expr> ID<ldent_nr> <Drive_type> ]
<axis_name>
Name of SERCOS axis
\begin{tabular}{|l|l|}
\hline Additional ID & Meaning \\
\hline ALL & Check all SERCOS axes existing in the system \\
\hline
\end{tabular}

AXNR<expr>
ID<Ident_nr>
<Drive_type>

Logical axis number of SERCOS axis, positive integer Identification number of the command in SERCOS format, e.g. S-0-0148 (drive-controlled referencing) or S-0-0170 (tracing cycle) If no identification number is programmed, the system waits for all open commands. Drive type
\begin{tabular}{|l|l|}
\hline Valid IDs & Meaning \\
\hline SERC & \begin{tabular}{l} 
SERCOS drive \\
(currently only one permitted)
\end{tabular} \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Synchronised wait (COMMAND)}
```

\#COMMAND WAIT SYN [AX=X, ID=S-0-0148, SERC]
\#COMMAND WAIT SYN [AX ALL, ID S-0-0148, SERC]

```

\section*{Attention}

No plausibility check is made for logical axis number or identification number. The operator is solely responsible for making the correct entries.
\(\qquad\)
\(\square\)
Notice
The motion is not forcibly stopped to wait for a SERCOS command. But if a SERCOS command is not terminated at the end of the motion block, no further NC block is processed and the motion is therefore stopped.

\section*{Programing Example}

The system waits until the S-0-0148 command terminates at the end of block N120.
```

N100 \#COMMAND WR SYN [AX Y ID S-0-0148 SERC]
N110 G01 X1000 F100
N120 \#COMMAND WAIT SYN [AX Y ID S-0-0148 SERC]
N130 G01 X2000

```

\section*{Attention}

A ("WAIT") command can only check commands which were previously started from the same processing level (decoding context or synchronous to processing at interpolation level). For example, a synchronised command ("SYN") can only be checked synchronously at interpolator level.

Programing Example

Although the command S-0-0148 may still be active, no wait is initiated at block N 110 because no active command S-0-0148 is detected at decoding level. The real state of the command at interpolation level is only detected at block N120.

N100 \#COMMAND WR SYN [AX Y ID S-0-0148 SERC]
N110 \#COMMAND WAIT [AX Y ID S-0-0148 SERC]
N120 \#COMMAND WAIT SYN [AX Y ID S-0-0148 SERC]

\subsection*{12.15 Channel synchronisation}

When a multi-channel controller is used (particularly in the case of \(n>2\) ), situations may occur in which it is absolutely essential to maintain with specific run sequences between channels.


Fig. 105: Example application: double-column machine with tool changer
The example in the figure above shows such an application in which two machining units (A and B) share a common machining space. Similarly, both machines use the tool changer resource (C). In order to avoid collisions in such a machine configuration, the NC programs of the various controller channels must be synchronised with each other at specific points. For example, in the above case, column A may not enter the collision space while column B is still located there. Similarly, column B may not use the tool changer if column A is currently accessing it.
For example, the time sequence in the two channels of the controller as shown in the figure below results from an access to the tool changer resource.


Fig. 106: Sequence in case of shared access to a resource
The required synchronisation is based on sending and waiting for signals and is performed by the NC commands in the NC programs described below.

\subsection*{12.15.1 Sending signals (\# SIGNAL)}

Basically, a distinction is made between signals with no receiver specified (also called broadcast signals) and non-broadcast signals where a channel is explicitly specified as the receiver.
The signals are identified by a unique number, although it is permitted to send signals with identical signal numbers.

In the case of non-broadcast signals, one or more NC channels must be explicitly specified as receivers. If several receivers are specified for one signal, this acts in the same way as the multiple transmission of the same signals to single channels.

Example

\section*{Sending signals}
```

\#SIGNAL [ID4711 CH1 CH2 CH3]
acts in the same way as
\#SIGNAL [ID4711 CH1]
\#SIGNAL [ID4711 CH2]
\#SIGNAL [ID4711 CH3]

```

These signals are only valid for a WAIT of the addressed receiver and are used up if the consumption counter COUNT is not specified for a WAIT from the receiver channel. If a consumption counter is specified, the same number of WAIT requests are possible until the signal is used up.

As opposed to this situation, broadcast signals can be received by a WAIT from any channel.
If no consumption counter COUNT is specified for broadcast signals, they are not used up by a WAIT. This means that they remain until they are explicitly cleared (see \#SIGNAL REMOVE). If a consumption counter COUNT is specified, the exact same number of WAIT requests is possible until the broadcast signal is used up as is the case with non-broadcast signals.
```

\#SIGNAL [<mode>] [ ID<sgn_nr> [COUNT<expr>] { P[<idx>]=<param>} {CH<chan_nr>} ] (non-

```
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{<mode>} & Synchronisation mode. Permitted identifiers: \\
\hline & Synchronisation at decoding level (initial state) For example, this synchronisation is required if it is necessary to synchronise to parameters or variables. \\
\hline & \begin{tabular}{l}
SYN Synchronisation at interpolator level. \\
This synchronisation is required for real-time requests, e.g. synchronisation of two machining units on a multi-column machine.
\end{tabular} \\
\hline ID<sgn_no> & Signal number; must be unique system-wide. Positive integer. \\
\hline COUNT<expr> & Consumption counter; defines how many times a signal can be called with \#WAIT. Positive integer. \\
\hline \(\mathrm{P}[<i d x>]=\) pparam> & Signal parameter. The real values <param> are entered in the parameter array of the signal according to the specified index. \\
\hline & <idx> Range for maximum possible number of parameters: \\
\hline CH<chan_no> & Number of the channel for which the signal is destined. 1...max. number of channels \({ }^{(2)}\) \\
\hline & If no channel number is specified, a so-called broadcast signal is sent to all available signal receivers in the system. \\
\hline
\end{tabular}
(1) see [6] [> 819]-6.45, (2) see [6] [> 819]-2.4

\section*{Programing Example}

\section*{Sending signals}
```

(Signal 812, synchronisation at DEC level, broadcast)
N500 \#SIGNAL [ID812]
(Signal 4711, synchronisation at DEC level, to channel 2)
N100 \#SIGNAL [ ID4711 CH2 ]
(Signal 4711, synchronisation at DEC level,
for 10 \#WAIT requests, broadcast)
N100 \#SIGNAL [ ID4711 COUNT10 ]
(Signal 815, synchronisation at interpolator level,
twice to channel 2 and once to 3)
N200 \#SIGNAL SYN [ ID815 CH2 CH2 CH3 ]
(Signal 911, synchronisation at decoder level, to channel 3)
(1St signal parameter V.A.MEAS.ACS.VALUE.X, 2nd signal parameter P200,
3rd signal parameter 94.4)
N260 P200 = 924
N300 \#SIGNAL [ IDP100 CH3 P[0]=V.A.MEAS.ACS.VALUE.X P[1]=P200 P[2]=94.4 ]

```

\subsection*{12.15.2 Removing (broadcast) signals (\#SIGNAL REMOVE)}

Signals are generally cleared after consumption by an assigned WAIT. In addition, non-broadcast signals are implicitly cleared for an NC reset of the receiver channel (see section Reset handling). Since broadcast signals are not cleared by a WAIT unless the consumption counter is specified, they must be cleared explicitly. An additional NC command exists for this purpose. This NC command can also be used to clear normal signals although in this case the identification number and the addressed channel must match.

If a single signal is specified for clearing, only one signal is cleared if more than one identical signals exist. However, when specifying a signal range [ID; IDMAX], i.e. including several identical signal numbers, all signals within this range are cleared.
\#SIGNAL REMOVE [<mode>] [ ID<sgn_nr> | IDMIN<sgn_nr> [IDMAX<sgn_nr>]
\(\left\{\mathrm{CH}<c h a n \_n r>\right\}\) ] (non-modal)
\begin{tabular}{|c|c|}
\hline \multirow[t]{3}{*}{<mode>} & Synchronisation mode. Permitted identifiers: \\
\hline & Synchronisation at decoding level (initial state) For example, this synchronisation is required if it is necessary to synchronise to parameters or variables. \\
\hline & SYN Synchronisation at interpolator level. This synchronisation is required in the case of real-time requests, e.g. synchronisation of two machining units on a multi-column machine \\
\hline ID<sgn_no> & Number of broadcast signal to be cleared. Positive integer. \\
\hline IDMIN<sgn_no> & First broadcast signal in a range to be cleared. Also alternative to ID<sgn_nr>. Positive integer \\
\hline IDMAX<sgn_no> & Last broadcast signal in a range to be cleared. Positive integer \\
\hline CH<chan_no> & Number of channel for which the signal to be cleared is intended. 1...max. number of channels \({ }^{(1)}\) \\
\hline & If no channel number is specified, the corresponding broadcast signal is cleared \\
\hline
\end{tabular}
(1) see [6] [ 819]-2.4

\section*{Programing Example}

Clearing (broadcast) signals
```

(Clear a broadcast signal 812, synchronisation at DEC level)
N500 \#SIGNAL REMOVE [ID812] or
\#SIGNAL REMOVE [IDMIN812]
(Clear a signal }812\mathrm{ to channel2, synchronisation at DEC level)
N500 \#SIGNAL REMOVE [ID812 CH2]

```
(Clear all broadcast signals in 812-820,
synchronisation at DEC level)
N500 \#SIGNAL REMOVE [IDMIN812 IDMAX820] or
\#SIGNAL REMOVE [ID812 IDMAX820]
(Clear all signals 812 to channel 1, synchronisation at DEC level)
N500 \#SIGNAL REMOVE [ID812 IDMAX812 CH1]
(Clear a broadcast signal 813, synchronisation at IPO level)
N600 \#SIGNAL REMOVE SYN [ID813]

\subsection*{12.15.3 Waiting for signals (\#WAIT)}

Analogously to sending signals, it is possible with the WAIT command to wait for a corresponding SIGNAL. A broadcast WAIT waits only for a broadcast SIGNAL with the same signal number. A WAIT synchronised at decoder/interpolator level uses up one separate SIGNAL in each case.
\#WAIT [<Modus>] [ ID<sgn_nr> \{ P[<idx>] = <param> \} \{ CH<chan_nr> \} [ AHEAD ] ] (non-modal)
\begin{tabular}{ll} 
<mode> & \begin{tabular}{l} 
Synchronisation mode. Permitted identifiers: \\
Synchronisation at decoding level (initial state) For example, this syn-
\end{tabular} \\
chronisation is required if it is necessary to synchronise to parameters or \\
variables.
\end{tabular}

After complete acknowledgement of the waiting condition (reception of all required signals), a check is made whether all programmed parameters were written. An error message is generated if id not the case.

\section*{Attention}

Signal parameters can only be evaluated at decoder level. This means, for example, a \#WAIT SYN \([\ldots \mathrm{P}[0]=\ldots]\) is not allowed.

Channel number from which a signal is expected.
1...max. number of channels \({ }^{(2)}\)

If no channel number is specified, the program waits for a broadcast signal from any user.
AHEAD Keyword for execution of a "flying" WAIT. Used to reduce waiting times because of the buffer effect of the look-ahead function (up to 70 blocks in advance). If synchronised at interpolator level, WAIT is output at once. As a result, the following acknowledgement check (SIGNAL) is flying, i.e. a change can be made immediately to the next motion block without interruption.
(1) see [6] [> 819]-6.45
(2) see [6] [〉 819]-2.4

\section*{Programing Example}
```

(Wait flag 4711, synchronisation at DEC level, SIGNAL 4711 from
any channel)
N200 \#WAIT [ID4711]
((Wait flag 815, synchronisation at interpolator level,
SIGNAL 815 from channels 2 and 3)
N100 \#WAIT SYN [ID815 CH2 CH3]
(Wait flag 911, synchronisation at decoder level, from channel 3)
(1st signal parameter V.P.SIGNAL, 2nd signal parameter P200)
N250 P100 = 911
N300 \#WAIT [IDP100 P[0]=V.P.SIGNAL P[1]=P200 CH3]
(The calculation below only takes place when)
(the signal is received)
N350 P20 = 10 * P200

```
\begin{tabular}{|c|c|}
\hline & Programing Example \\
\hline \ll / > & Wait for signals with adoption of parameters (in channel 3): \\
\hline
\end{tabular}
\%channel1
N10 \#SIGNAL [ID \(110014 \mathrm{P}[0]=1234 \mathrm{CH} 3]\)
N20 M30
\%channel2
N10 \#SIGNAL [ID 110014 P[1] = 200 CH3]
N20 M30
\%channel3
N10 P1 = 1 (Stores value from channel 1
N20 P2 = 1 (Stores value from channel 2
N30 XP1 YP2
N40 \#WAIT [ID \(110014 \mathrm{P}[0]=\mathrm{P} 1 \mathrm{P}[1]=\mathrm{P} 2 \mathrm{CH} 1 \mathrm{CH} 2]\)
N50 XP1 YP2
N60 M30
kernel Industrielle Steuerungstechnik GmbH

\subsection*{12.15.4 Reading signals without waiting (\#SIGNAL READ)}

\section*{Release Note}

This function is available as of CNC Build V2.11.2820.00

The NC command \#WAIT stops the program decoding of the interpreter if the requested signal is missing. This blocks further program processing if the signal never arrives.

The NC command \#SIGNAL READ allows flexible program sequences without a program stop. This command stores the result of the signal read operation in the channel-specific variable V.G.SIGNAL_READ. When this variable is subsequently evaluated, a corresponding response is possible.
\#SIGNAL READ behaves analogously to \#WAIT with respect to the use of signals as well as the programming and use of parameters and broadcast signals.

\section*{Attention}

The NC command \#SIGNAL READ is only permitted at interpreter level. A \#SIGNAL READ SYN [..] is not permitted and is indicated by an error message.
```

\#SIGNAL READ [ ID<sgn_nr> { P[<idx>] = <param> } { CH<chan_nr> } ] (non-modal)

```
\begin{tabular}{|c|c|}
\hline ID<sgn_no> & Number of the signal to be read. Positive integer. \\
\hline \multirow[t]{3}{*}{\[
\mathrm{P}[<i d x>]=<p a r a m>
\]} & Signal parameter as real number. When signals are read, parameters can also be sent by the signal sender. Parameters can also originate from different channels. They are assigned to the specified parameters or variables (<param>). \\
\hline & \(<i d x>\quad \begin{aligned} & \text { Range for maximum possible number of parameters: } 0 . . . \text { max. Number of } \\ & \text { signal parameters }{ }^{(1)}\end{aligned}\) \\
\hline & After complete acknowledgement of all the required signals, a check is made whether all programmed parameters were written. If this is not the case, the program stops with an error message. \\
\hline \multirow[t]{2}{*}{CH<chan_no>} & Channel number from which a signal is expected. 1...max. number of channels \({ }^{(2)}\) \\
\hline & If no channel number is specified, the program waits for a broadcast signal from any user. \\
\hline \multicolumn{2}{|r|}{(1) see [6] [> 819]-6.45} \\
\hline & [> 819]-2.4 \\
\hline
\end{tabular}

The status of the read access of \#SIGNAL READ is indicated by the variable V.G.SIGNAL_READ. It is TRUE if the corresponding signal was present. The value of the variable remains until the next read access with \#SIGNAL READ.
\begin{tabular}{ll} 
V.G.SIGNAL_READ & (Status of read access of \#SIGNAL READ) \\
& (TRUE: Signal present and read) \\
& (FALSE: No signal present, default)
\end{tabular}

\section*{Programing Example}

Read signals and wait for result
```

(Wait flag 4711, synchronisation at interpreter level, SIGNAL 4711 from any channel)
:
N100 \#SIGNAL READ [ID=4711]
N110 \$IF V.G.SIGNAL_READ == TRUE
N120 LL UP1
N130 \$ELSE
N140 \#ERROR [..]
N150 \$ENDIF
:
(Wait flag 815, synchronisation at interpreter level,
SIGNAL 815 from channel 1)
:
N100 \#SIGNAL READ [ID=815 CH=1]
N110 \$IF V.G.SIGNAL_READ == TRUE
N120 LL UP1
N130 \$ELSE
N140 L Init.nc
N150 \$ENDIF
:
(Wait flag 911, synchronisation at interpreter level, from channel 3)
(1st signal parameter V.P.SIGNAL, 2nd signal parameter P200)
:
N100 \#SIGNAL READ [ID=911 P[0]=V.P.SIGNAL P[1]=P200 CH=3]
N110 \$IF V.G.SIGNAL_READ == TRUE
N120 P20 = [10 * P200]-V.P.SIGNAL
N130 \$ELSE
N140 P20 = 0
N150 \$ENDIF

```

\subsection*{12.15.5 RESET handling}

If a single channel is reset, the synchronisation events of the channel affected are cleared, i.e. all wait requests (\#WAIT) which the channel in question has sent and all non-broadcast signals (\#SIGNAL) destined for it are cleared.

Broadcast signals are not cleared if a channel is reset since these signals may still be expected by channels. They must be cleared explicitly (\#SIGNAL REMOVE).

\subsection*{12.15.6 Synchronisation scenarios}

\section*{Synchronisation of 2 decoders on 2 channels}


Fig. 107: Synchronisation of 2 decoders on 2 channels
- Decoder 3 waits for decoder 1, decoder 1 continues to operate without interruption

\section*{Programing Example}

\section*{Synchronisation of 2 decoders on 2 channels}
```

% kanal_1
...
(Signal P100)
(Synchronisation at DEC level)
(Synchronisation with channel 3)
(Parameter V.P.SYNC)
V.P.SYNC = 1000
P100 = 814
\#SIGNAL [IDP100 P[0]= V.P.SYNC CH3]

```
. . .
```

% kanal_3
...
(Wait request 814)
(Synchronisation at DEC level)
(Synchronisation with channel 1)
(Parameter V.P.SIGNAL)
\#WAIT [ID814 P[0]= V.P.SIGNAL CH1]

```
...

\section*{Synchronisation between decoder and interpolators on 3 channels}


Fig. 108: Synchronisation between decoder and interpolators on \(\mathbf{3}\) channels
- Interpolator 1 waits for interpolator 2 and decoder 3,
- Interpolator 2 waits for interpolator 1 and decoder 3,
- Decoder 3 signals to interpolator 1 and interpolator 2.

\section*{Programing Example}

Synchronisation between decoder and interpolators on 3 channels
```

% kanal_1
...
(Wait request 968)
(Sync. At IPO level)
(Sync. with channels 2 and
3)
\#WAIT SYN [ID968 CH2 CH3]

```
```

%kanal_2
(Wait request 968)
(Sync. At IPO level)
(Sync. with channels 3
and 1)
\#WAIT SYN [ID968 CH3 CH1]

```
```

% kanal_3
(Signal 968)
(Sync. At DEC level)
(Sync. with channels 1 and
2)
\#SIGNAL [ID968 CH1 CH2]

```

\section*{Synchronisation between interpolators on 3 channels}


Synchronisation between interpolators on 3 channels
- Interpolator 1 waits for interpolator 2,
- Interpolator 3 waits for interpolator 2,
- Interpolator 2 signals to interpolator 1 and interpolator 3.

Programing Example
Synchronisation between interpolators on three channels
\begin{tabular}{|c|c|c|}
\hline \% kanal_1 & \%kanal_2 & \% kanal_3 \\
\hline (Wait request 100) & (Signal 100) & (Wait request 100) \\
\hline (Sync. At IPO level) & (Sync. At IPO level) & (Sync. At IPO level) \\
\hline (Sync. with channel 2) & (Sync. with channels 1 and 3) & (Sync. with channel 2) \\
\hline \#WAIT SYN [ID100 CH2] & \#SIGNAL SYN [ID100 CH1 CH3] & \#WAIT SYN [ID100 CH2] \\
\hline
\end{tabular}

\section*{Synchronisation between decoder and interpolator of one channel}


Fig. 109: Synchronisation between decoder and interpolator of one channel
- Decoder waits until interpolator has reached position X 250.
- Motion block "G01 X370 Z200 F80" is already on the NC channel and is processed after signalling.
- Motion block "G01 X900" is decoded only after synchronisation.

\section*{Attention}

In the case of synchronisation requests between decoder and interpolator, states may occur in which the NC program cannot be decoded further since no acknowledgement has yet arrived. The acknowledgement is, however, not dispatched by the interpolator since the signal block does not reach the interpolator owing to the buffer effect of the NC channel. A \#FLUSH which flushes the NC channel must be provided in such cases in order to avoid possible deadlocks.

\section*{Programing Example}

Synchronisation between decoder and interpolator of one channel
```

% kanal_1
G00 X100 Y500
G01 X250 F300
(Signal 88)
(Synchronisation at IPO level)
(Synchronisation with channel 1)
\#SIGNAL SYN [ID88 CH1]
(Operation)
G01 X370 Z200 F80
(Wait request 88)
(Synchronisation at DEC level)
(Synchronisation with channel 1)
\#FLUSH
\#WAIT [ID88 CH1]
G01 X900

```

\subsection*{12.16 Rotate the coordinate system in the plane (\#ROTATION ON/OFF)}

This function rotates a coordinate system in the current plane (G17/G18/G19). Contours programmed in the machine coordinate system can be adapted quickly and easily to workpieces in offset positions.
Contour rotation acts directly on the programmed axis coordinates (contour) before all other con-tour-influencing functions, i.e. all offsets and mirroring operations are not influenced by the rotation and can be used as before (*).
Rotation may also be applied within an already rotated coordinate system (\#(A)CS).
A change of plane with \(\mathrm{G} 17 / \mathrm{G} 18 / \mathrm{G} 19\) automatically deselects an active contour rotation and a warning is output.
As a alternative to \#ROTATION, contour rotation can be programmed using G68/G69 [ 190].

Syntax:
```

\#ROTATION ON [ [ [ANGLE=..][CENTER1=..][CENTER2=..] ]] modal
\#ROTATION OFF modal

```

ANGLE=..
CENTER1=..
CENTER2=..

Rotation angle in [ \({ }^{\circ}\) ]
Offset of the first main axis relative to the centre of rotation in [mm, inch]
Offset of the second main axis relative to the centre of rotation in [mm, inch]


Fig. 110: Significance of rotation parameters in the main plane (example G17):
\begin{tabular}{|l|l|l|}
\hline a: CENTER1 & b: CENTER2 & c: ANGLE \\
\hline
\end{tabular}

The programmed rotation parameters can be read with the following variables:
V.G.ROT_ACTIVE Supplies the value 1 if a rotation is active
V.G.ROT_ANGLE Rotation angle
V.G.ROT_CENTER1 Offset of the first main axis relative to the centre of rotation
V.G.ROT_CENTER2 Offset of the second main axis relative to the centre of rotation

\section*{Notice}
(*) It makes no difference whether the offsets (e.g. G54, G92 etc. ) were programmed before or after the \#ROTATION command; they always act in the axis directions of the basic coordinate system of the machine (MCS).
In addition, tool offsets always act independently of P-TOOL-00010 in the axis directions of the MCS.

\section*{Programing Example}

Rotation in a plane (contour rotation)
\%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29
\%ang1.nc
N100 G53 G17
N110 LL part
N130 \#ROTATION ON [ANGLE -45 CENTER1=10 CENTER2=100]
N140 LL part
N150 G21 (mirror X coordinates)
N160 LL part
N170 G18 (warning expected)
N190
M30


Same contour as in the previous program but within \#CS of \(-15^{\circ}\).
```

%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

```
\% ang1cs.nc
N99 \#CS ON[0,0,0,0,0,-15]
N100 G53 G17
N110 LL part
N130 \#ROTATION ON [ANGLE -45 CENTER1 10 CENTER2 100]
N140 LL part
N150 G21 (mirror X coordinates)
N160 LL part
N190 \#CS OFF
M30

```

%L Trajectory0
N10 G54 G90 X0 Y0
N20 G0 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G0 X75
N70 X0 Y0
N80 M29
%ang2.nc
F1000
N100 LL Trajectory0
N200 G92 G90 Y-25
N400 \#ROTATION ON [ANGLE 90 CENTER1 }75\mathrm{ CENTER2=-50]
N600 LL Trajectory0
N700 G92 G90 Y25
N900 \#ROTATION ON [ANGLE=-90 CENTER1 75 CENTER2 50]
N60 LL Trajectory0
N70 M30

```

```

%L Trajectory0
N10 G54 G90 X0 Y0
N20 G1 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G1 X75
N70 X0 Y0
N80 M29

```
\%ang3.nc
N10 F4000 G90
N15 \#ROTATION ON
N20 LL Trajectory0
N30 G90 G92 Y100
N35 \#ROTATION ON [ANGLE 180]
N40 LL Trajectory0
N50 G90 G92 Y-100
N55 \#ROTATION ON [ANGLE 180]
N60 LL Trajectory0
N70 M30

```

%L UPRG1
N1 X0 Y0 Z0
N10 X25
N30 X0
N40 Y25
N50 Y0
N60 X10
N70 Y10
N80 X0 Y0
N90 Y10
N100 X10 Y0
N110 G03 I-5 J5 Y10
N120 G1 X0 Y0
M17
%L UPRG2
N2 X0 Y0 Z0
N10 X25
N20 G02 I0.8
N30 G1 X0
N40 Y25
N45 G02 J0.8
N50 G1 Y0
N120 G1 X0 Y0
M17
%L UPRG3
N3 G1 X0 Y0 Z0
N10 X4 Y4
N20 G02 I1 J1
N30 G1 X0 Y0 Z0
M17
%ang4.nc
N1 G1 X0 Y0 Z0 F1000
N500 G92 X10 Y10
N510 LL UPRG1
N520 \#ROTATION ON [ANGLE O CENTER1 25 CENTER2 15]
N540 LL UPRG1
N550 G92 X20 Y25
N560 \#ROTATION ON [ANGLE -35]
N570 LL UPRG1
N580 G92 X35 Y-10
N590 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG1
N610 \#ROTATION ON [CENTER1 O CENTER2 O]
N620 LL UPRG1
N630 \#ROTATION ON [ANGLE=V.G.ROT ANGLE+117]
N640 LL UPRG1
N650 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
N500 G92 X10 Y10
N510 LL UPRG2
N520 \#ROTATION ON [ANGLE O CENTER1 25 CENTER2 15]
N540 LL UPRG2
N550 G92 X20 Y25
N560 \#ROTATION ON [ANGLE -35]
N570 LL UPRG2
N580 G92 X35 Y-10
N590 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG2
N610 \#ROTATION ON [CENTER1 O CENTER2 O]

```
```

N620 LL UPRG2
N630 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
N640 LL UPRG2
N650 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
N500 G92 X10 Y10
N510 LL UPRG3
N520 \#ROTATION ON [ANGLE O CENTER1 25 CENTER2 15]
N540 LL UPRG3
N550 G92 X20 Y25
N560 \#ROTATION ON [ANGLE -35]
N570 LL UPRG3
N580 G92 X35 Y-10
N590 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG3
N610 \#ROTATION ON [CENTER1 O CENTER2 O]
N620 LL UPRG3
N630 \#ROTATION ON [ANGLE=V.G.ROT ANGLE+117]
N640 LL UPRG3
N650 \#ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
M30

```


Test of relative and absolute programming:
```

%L contour 1
N1 G1 G91 (all positions with G91)
N2 X20
N3 Y20
N4 X20
N5 Y20
N6 X20
N7 Y20
N8 X20
N9 Y2O
N10 X20
N11 Y20
N12 X5
N13 Y-3
N14 Y3
N15 X-5
N16 G90 X0
N17 Y0
N18 X5
N19 Y-3
N20 Y0
N21 X0
\#MSG SYN["contour_1 finished"]
M17
%L contour 2
N100 G1 (same contour, X with G91, Y with G90)
N101 G91 X20
N102 G90 Y10 (transl. offset in Y is 10)
N103 G91 X20
N104 G90 Y30
N105 G91 X20
N106 G90 Y50
N107 G91 X20
N108 G90 Y70
N109 G91 X20
N110 G90 Y90
N111 G91 X8
N112 G91 Y-4
N113 G91 Y4
N114 G91 X-8
N115 G90 X0
N116 Y0
N117 X8
N118 Y-4
N119 Y0
N119 Y0
N120 X0
\#MSG SYN["contour_2 finished"]
M17
%L contour 3
N200 G1 (same contour, Y with G91, X with G90)
N201 G90 X0 (transl. offset in X is 20)
N202 G91 Y20
N203 G90 X20
N204 G91 Y20
N205 G90 X40
N206 G91 Y20
N207 G90 X60
N208 G91 Y20
N209 G90 X80
N210 G91 Y20

```
```

N211 G91 X11
N212 G91 Y-5
N213 G91 Y5
N214 G91 X-11
N215 G90 X0
N216 Y0
N217 X11
N218 Y-5
N219 Y0
N220 X0
\#MSG SYN["contour_3 finished"]
M17
%L contour 4
N300 G1 G90}\mathrm{ (same contour with G90)
N301 X0 (transl. offset in X is 20)
N302 Y10 (transl. offset in Y is 10)
N303 X20
N304 Y30
N305 X40
N306 Y50
N307 X60
N308 Y70
N309 X80
N310 Y90
N111 G91 X14
N312 G91 Y-6
N313 G91 Y6
N314 G91 X-14
N315 G90 X0
N316 Y0
N317 X14
N318 Y-6
N319 Y0
N320 X0
\#MSG SYN["contour 4 finished"]
M17
%ang5.nc
N5001 G0 G90 X0 Y0 F5000

```
N501 \#ROTATION ON [ANGLE 0 CENTER1 20 CENTER2 10]
    (Note: with angle != 0 the contours are
    (not congruent because of difference
    (of absolute and incremental movement!)
N502 \#ROTATION ON
N503 LL contour 1
N504 \#ROTATION ŌFF
N5002 G0 G90 Y0
N5003 X0
N505 \#ROTATION ON
N506 LL contour 2
N507 \#ROTATION ŌFF
N5004 G0 G90 Y0
N5005 X0
N508 \#ROTATION ON
N509 LL contour 3
N510 \#ROTATION OFF
N5006 G0 G90 Y0
N5007 X0

N511 \#ROTATION ON

N512 LL contour 4
N513 \#ROTATION OFF
N5005 G0 G90 Y0
N5006 X0
N210 M2


After selecting valid rotation angular. The offsets for the rotation point may firstly be considered with the first absolute programming (G90).
\%ang6.nc
N10 G90 X0Y0Z0 G1 F200
N20 \#ROTATION ON [ANGLE 30 CENTER1 10 CENTER2 10]
N30 X0 Y0
N40 G3 I1 J1 F500
N50 G01 X10
N60 Y10
N70 G90 X0
N80 G90 Y0
N90 X10 Y10
(New rotation parameters.
(Note: Centre offset has no effect until an absolute (G90) position
(has been programmed. However, the angle is effective.
N100 \#ROTATION ON [ANGLE 10 CENTER1 -10 CENTER2 0]
N110 G3 I1 J1 F500
N120 G01 G91 X10
N130 G91 Y10
N140 G91 X-10
N150 G91 Y-10
(Make the new centre effective by first absolute position:
N200 G90 X0 Y0
N210 G3 I1 J1 F500
N220 G01 X10
N230 Y10
N240 X0
N250 Y0
M30


Transforming the absolute or relative programmed circle centre point:
\%ang cent.nc
N10 F2000 G53
N11 G0 X0 Y0 G90
```

(------------------------------------------------------------------------------

```
(4 times the same circle with different programming of circle centre point)


N12 G0 X0 Y0 G90
N13 Y50
N14 X-75
N15 G3 X-75 Y-50 G161 I-75 J0 (absolute centre)
N16 G0 X0 Y0 G90
N17 Y50
N18 X-75
N19 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N20 G0 X0 Y0 G90
N28 \#ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 -75]

(The same with LIN and ANG offset active (ED=0))

N80 G0 X0 Y0 G90
N90 Y50
N100 X-75
N110 G3 X-75 Y-50 G161 I-75 J0 (absolute centre)
N120 G0 X0 Y0 G90
N130 Y50
N140 X-75
N150 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N360 G0 X0 Y0 G90

(The same rotated by \(50^{\circ}\) (unnecessary I / J omitted))

N370 \#ROTATION ON [ANGLE 50]
N380 G0 X0 Y0 G90
N390 Y50
N400 X-75
N410 G3 X-75 Y-50 G161 I-75 (J0) (absolute centre, not prog. is 0)
N420 G0 X0 Y0 G90
N630 Y50
N640 X-75
N650 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N655 G0 X0 Y0 G90

(The same rotated by \(95^{\circ}\) )

N660 \#ROTATION ON [ANGLE 95]
N670 G0 X0 Y0 G90
N680 Y50
N690 X-75
N700 G3 X-75 Y-50 G161 I-75 J0 (absolute centre)
N710 G0 X0 Y0 G90
N730 Y50
N740 X-75
N750 G3 X-75 Y-50 G162 I0 J-50 (relative centre)
N760 G0 X0 Y0 G90
M30


\subsection*{12.17 Automatic axis tracking (C axis tracking) (\#CAXTRACK)}

Machining operations, such as cutting various materials, require tool guidance generally associated with the C axis so that the tool is always aligned tangentially relative to the path followed.

It must be noted that the tangent is not unique at each point of the path (break points). Consequently, a solution requires strategies for handling block transitions which do not feature constant tangents.

One typical application is the technology sector for glass cutting. This involves machining level contours with the aid of cutting tools in the form of carbide metal cutting wheels using CNC machines. The flat workpiece is scored slightly at the machining point in accordance with the programmed contour (closed contour, e.g. ellipse). The required contour can then be broken out of the glass workpiece.


Starting point
C-axis in basic orientation

Fig. 111: Tracking the rotary C axis tangentially relative to the \(x-y\) contour

The C axis can also be guided tangentially relative to the path by explicit programming. However, the NC commands described below simplify programming considerably.
```

\#CAXTRACK ON [ [ [ANGLIMIT=..] [OFFSET=..] [OPTALIGN=..]
[ROTMODE=..] [SCALEFACT=..]
[AX=.. | AXNR=..]
[START_STROKE] [SYMMETRIC_TOOL]]

Limit angle in [ ${ }^{\circ}$ ].
This parameter is only considered for non-tangential continuous contour sections. For example, tangential continuous contour sections are created by the contouring function G61.

If the angle between the tangents to the contour exceeds the limit angle on block transition, the contouring motion is halted and the dressing movement is performed by an inserted motion block at rapid traverse velocity. In this case, the inserted movement forms one unit with the following second block. This means in particular that PLC synchronisation events in conjunction with the motion (M functions etc.) are possible only before or after this motion unit.

If the transition angle to second block is smaller than the limit angle, the dressing motion starts immediately on transition to the second block. In general limited axis acceleration in general causes a lower feedrate at the block transition. If this reaction is not acceptable, the axis dynamic monitoring of the tracking axis can be excluded. (e.g. G116 C1, see Sec. 4.31 [ 185])

OFFSET=

OPTALIGN=..

Angle offset in [ ${ }^{\circ}$ ].
This specifies an angle offset in order to orient the tool opposite to the tangent against the contour.
When selected, orientation is automatically optimised if the orientation distance is greater than the defined angle value.
This parameter is only considered for active automatic aligning according to P -CHAN-00101 and rotary linear axes with limited motion range (no modulo axis). It is only effective during the automatic orientation process on the first contour element.
After selecting the tangential tracking function with automatic orientation, the position of the tracking axis lies without any offset in the range of -180 to $+180^{\circ}$. The parameter allows that the position of the tracking axis is considered during the automatic orientation process before the tangential tracking function is selected.

The function is only useful if the tracking axis has approximately the correct position to the first contour element before automatic orientation is selected. However, for example, it has a pre-orientation of $+-360^{\circ}$. If the internally calculated orientation angle exceeds the programmed angle 'OPTALIGN', alternative solutions for the orientation angle are taken into consideration. Then the smallest orientation distance of the solutions determines the real orientation angle. (*)

## Notice

$\left(^{*}\right)$ For modulo axes, the automatic orientation process is always executed on the shortest way.

| ROTMODE=.. | Boolean value indicating assignment of the tracking axis: <br> 0: The tracking axis is an axis in the tool (default). |
| :--- | :--- |
|  | 1: The tracking axis is an axis in the workpiece. |
| The tool axis always has to be aligned vertically to the XY plane. Alternatively, the po- |  |
| sition of the tracking axis can also be defined in the channel parameters (P- |  |
| CHAN-00185). |  |

## Notice

The tracking axis used must be an additional axis. This may not lie on a main axis index.

Automatic tracking of the axis is performed with the correct sign relative to the last position depending on the resulting contour transition angle.

## Notice

Depending on the P-CHAN-00101 parameter, the alignment of the tracking axis in the required orientation (generally parallel to the contour) is as follows:

- Programmed aligning before selecting automatic tracking. No check of the position is conducted. The current angle position is frozen and the tracking axis is aligned at that angle.
- Automatic tangential alignment of the tracking axis (P-CHAN-00101) to the first programmed contour element when automatic tracking is selected.

Caution: The activation of automatic tangential alignment is absolutely necessary if there are programmed contour sections with polynomials (e.g. G261).

Tracking commences with the transmission of the first to the second relevant motion block after i is activated by \#CAXTRACK ON. Automatic axis tracking operates in the main plane of circular interpolation (1st + 2nd main axes). This must be defined before activation (G17 / 18 / 19, \#PUT AX / \#CALL AX / \#SET AX).

If the tracking axis already has the correct orientation when selected, the path motion is continued without interruption to the first relevant motion block provided by the parameter P-CHAN-00109.

If the contouring function (G261) is already active and parameter ANGLIMIT $>0$, the following condition is required for a smooth motion transition:

- Contour elements before and after \#CAXTRACK ON [..] must be tangential linear blocks to each other.


## Notice

Do not operate an active tracking axis in synchronous mode as a slave axis.

## Programing Example

## Automatic axis tracking (C axis tracking)

```
Example 1: Selecting axis tracking
N10 G00 G90 X0 Y0 Z0 CO
N20 X5 Y5 C45 ;straight line 45' to the X axis, tracking axis C
    ;aligned parallel to the contour
N20 #CAXTRACK ON [ANGLIMIT 3, OFFSET O] ; Enable axis
                                ;tracking,limit angle 3*
    ;Angular offset 0*
N30 X10 Y10 ;Primary motion block, C axis is
    ;already aligned
N40 X20 ; Angle to previous block: -45* >
    ;Limit angle -> Block is inserted: End
    ;position of C = 0
N50 M99 X30 ;If M function synchronisation before
    ;motion > First synch., then motion C
    ; at 0, then X at 30
    ;If sync. After block > motion C at 0
    ;then X at 30, then sync.
N60 X40 ;C axis angle 0*
N70 X30 ;C axis angle 180
N80 Y0 ;C axis angle -90*
N90 #CAXTRACK OFF ;Disable axis tracking
M30
```

Example 2: Couple a slave axis (C2) to master tracking axis (C)
N20 G00 X0 YO Z0 CO A0 C2=0 A2=0
N50 \#SET AX LINK[1, C2=C]
N70 \#ENABLE AX LINK[1]
N140 G01 X0 Y0 Z0 A0 C0 F2000
N170 \#CAXTRACK ON [AX=C ANGLIMIT 0.1]
N190 LL SUB 1
N220 \#CAXTRACK OFF
N250 \#DISABLE AX LINK[1]
M30

### 12.18 User-defined error output (\#ERROR)

The NC command \#ERROR allows the output of user-defined error messages which are further processed by the higher-level GUI (GUI = Graphical User Interface). Additional parameters offer the option of specifying the error more precisely.
\#ERROR [ [ID<expr>] [RC<expr>] [MID<expr>] \{PV<|><expr>\} \{PM<|><expr>\}

$$
\begin{equation*}
\{\mathrm{PIV}<i><e x p r>\}] \tag{nonmodal}
\end{equation*}
$$

## ID<expr> <br> Error number:

| Valid values | Meaning |
| :---: | :--- |
| $1 . .1000$ | The numerical value determines the user-specific error number to be <br> output. |

$\mathrm{RC}<e \operatorname{expr}>\quad$ Error remedy class:

| Valid values | Meaning |
| :---: | :--- |
| $\mathbf{0}$ | Warning: |
| $\mathbf{2}$ | No transition to error state. Program execution is continued. |
| $\mathbf{7}$ | Error: |
| Transition to error state. Can be cleared with NC-RESET. |  |
| $\mathbf{7}$ | Fatal error: |
| Transition to 'system error' state. Requires restart of controller. |  |

MID<expr>

PV<i><expr>

PM<i><expr>

Multiple ID. Counter acts as a distinguishing feature if the \#ERROR command with the same error number (ID) is used several times in an NC program. MID must be a positive integer.
Max. 5 (1 <= i<=5) user-specific numerical values (PV1...PV5) in real format can also be output in the error message.
The maximum of $5(1<=\mathrm{i}<=5)$ PM parameters (PM1...PM5) specify the meaning of the PV parameters more precisely.

| Valid values | Meaning |
| :---: | :--- |
| $\mathbf{0}$ | IGNORE, value has no meaning |
| $\mathbf{1}$ | Limit value |
| $\mathbf{2}$ | Current value |
| $\mathbf{3}$ | Error value |
| $\mathbf{4}$ | Expected value |
| $\mathbf{5}$ | Corrected value |


| 6 | Logical axis number |
| :--- | :--- |
| 7 | Drive type |
| 8 | Logical control element number |
| 9 | State |
| 10 | Transition |
| 11 | Sender |
| 12 | Class |
| 13 | Instance |
| 14 | Identification number |
| 15 | Status |
| 16 | Uing number |
| 18 | Initial value |
| 19 | Final value |
| 20 |  |
| 21 |  |

PIV<i><expr>
The maximum of 4 ( $1<=\mathrm{i}<=4$ ) PIV parameters (PIV1...PIV4) transfer additional information in real format.

For non-programmed parameters, the following default values are valid:

| ID | 1 |
| :--- | :--- |
| RC | 0 |
| MID | 0 |
| PV1...PV5 | 0.0 |
| PM1...PM5 | 1 |
| PIV1...PIV4 | 0.0 |

## Programing Example

## User-defined error output

```
; -------
; Output of warning with ID 100, multiple identifier 10
#ERROR [ID100 RC0 MID10]
; ..
; -------
; Output of warning with ID 455 with parameter
; Error 455 with parameters
; Parameter 1 - current value is 1
; Parameter 1 - incorrect value is 4.999
#ERROR [ID455 RC2 PV1=5 PV2=4.999 PM1=2 PM2=3]
; ..
; -------
; Fatal error 999
#ERROR [ID999 RC7]
```


### 12.19 Time measurement (\#TIMER)

The NC command \#TIMER offers the option of time measurement in the NC program. The time recorded is represented in the unit milliseconds (ms).

## Attention

Time counters are provided channel global. For example, this permits the measurement of signal propagation times between channels.
For parallel independent time measurements in different channels, make sure that different counter numbers (IDs) are used. Otherwise, the measurements will influence each other.

```
#TIMER <Action> [<Mode>] [ID<counter_nr>]

Determines the action with the designated counter (ID).
\begin{tabular}{|c|l|}
\hline Action & Meaning \\
\hline START & Starts the designated counter (ID). \\
\hline STOP & Stops the designated counter (ID). \\
\hline READ & \begin{tabular}{l} 
Reads out the designated counter (ID). \\
The timer count is latched and saved to the assigned V.G.TIMER[ID] \\
variable in milliseconds (ms).
\end{tabular} \\
\hline CLEAR & \begin{tabular}{l} 
Resets and stops the counter (ID). \\
The assigned V.G.TIMER variable is not deleted but is retained until \\
the next READ action of the related counter.
\end{tabular} \\
\hline
\end{tabular}

\section*{Attention}

The timer function records a maximum of 1193 hours.
<Mode> Synchronisation mode:
\begin{tabular}{|c|l|}
\hline Valid modes & Meaning \\
\hline--- & \begin{tabular}{l} 
Time measurement asynchronous relative to the interpolator at decod- \\
ing level (initial state). Time measurement starts directly after decoding.
\end{tabular} \\
\hline SYN & \begin{tabular}{l} 
Time measurement at interpolator level. The designated counter is set \\
synchronous to the machining operations of the NC machine. The syn- \\
chronous read function \((<\) SYN \(>)\) in the interpolator interrupts decoding \\
until the timer count at decoder level is adopted in the timer variable.
\end{tabular} \\
\hline
\end{tabular}
kernel Industrielle Steuerungstechnik GmbH

\section*{Notice}

ID<counter_nr> Counter number:
\begin{tabular}{|c|l|}
\hline Valid values & Meaning \\
\hline \(0 \ldots 127\) & \begin{tabular}{l} 
A maximum of 128 channel global counters can be programmed. \\
However, only one counter (ID) can be programmed per timer com- \\
mand.
\end{tabular} \\
\hline
\end{tabular}

\section*{Programing Example}

Time measurement
```

:
\#FILE NAME[ MSG="C:\timer.txt" ] File name for time logging
:
\#TIMER START [ID=10] Start timer 10 (decoder level)
\#TIMER START SYN [ID11] Start timer 11 (IPO level)
:
:
\#TIMER READ [ID10] Save timer count in V.G.TIMER[10]
\#TIMER READ SYN [ID11] Save timer count in V.G.TIMER[11]
\#MSG SAVE["T10 = %d",V.G.TIMER[10]] Log timer count to file
\#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
\#TIMER STOP [ID10] Stop timer 10
\#TIMER CLEAR [ID10] Reset timer 10
:
:
\#TIMER READ SYN [ID11] Save timer count in V.G.TIMER[11]
\#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
:
:
\#TIMER READ SYN [ID11] Save timer count in V.G.TIMER[11]
\#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
:
:
\#TIMER READ SYN [ID11] Save timer count in V.G.TIMER[11]
\#MSG SAVE["T11 = %d",V.G.TIMER[11]] Log timer count to file
\#TIMER STOP SYN [ID11] Stop timer 11
\#TIMER CLEAR SYN [ID11] Reset timer 11
:
:
\#TIMER START [ID=10, ID11] Error, only one counter per timer
command permissible!

```

\subsection*{12.20 \\ Definition of feed axes (\#FGROUP, \#FGROUP ROT, \#FGROUP WAXIS)}

The parameter \#FGROUP defines the axes to which the programmed feedrate (F word) refers. For axes programmed in the command \#FGROUP a path in space is defined for execution at the programmed feedrate. All other axes are treated as tracking axes and they then reach their target points simultaneously with the path axes.

A characteristic feature of a path axis is that the distance to be moved is included in the feedrate. On the other hand, the tracking axis distance to be moved has no direct influence on path velocity.
```

\#FGROUP <Achsname> {,<Achsname>} ] (modal)

```
<Achsname>
Name of the axes which are members of the feed group
\#FGROUP (modal)

If no feed axes are programmed, the default settings of the channel parameters P-CHAN-00096 and P -CHAN-00011 are valid. If no feed axes are configured there, all main axes (Index \(0,1,2\) ) automatically form the channel feed group. This is indicated by the message 21209.

With linear interpolation, feed axes may be given any definition:
With circular and polynomial interpolation, the following exceptions apply:
- With circular interpolation, either all main axes must be feed axes or all defined feed axes must be tracking axes.
- With polynomial interpolation, all main axes form the feed group independent of the \#FGROUP command. An exception to this is polynomial contouring in DIST_SOFT mode. Here the programmed \#FGROUP is effective.

\section*{Programing Example}
```

N10 \#FGROUP [X,Y]
:
N100 \#FGROUP
: N999 M30

```
N50 \#FGROUP [A] (Tracking axis A is feed axis)
```

(Feed axes acc. to the default settings)
(in the channel parameters)
(X and Y are feed axes)

```

In order to machine cylindrical workpieces on a rotary workpiece axis, the real programmed feedrate [ \(\mathrm{mm} / \mathrm{min}\) ] should act at the tool contact point. This can be ensured either by selecting a suitable kinematic transformation (e.g. lateral surface transformation) or by using the command \#FGROUP ROT[...]. After programming this command, the feedrate of the rotary axis in [ \(\% / \mathrm{min}\) ] is recalculated depending on the reference radius. When the rotary axis is programmed alone or together with linear axes, the required programmed feedrate is obtained at the reference radius.

\section*{\#FGROUP ROT [ AX<Achsname> REF<Bezugsradius> ]}
\(A X<A c h s n a m e>\quad\) Name of the axis on which the reference radius is to act.
REF<Bezugsradius> Effective radius of the rotary axis in [mm, inch].
Feedrate calculation relative to a rotary axis is deselected by:
\#FGROUP ROT (modal)

\section*{Attention}

No check is made whether the axis "AX.." is really a rotary axis or not.
The function can only be used for feed blocks (G01) and combined with G94.

\section*{Notice}

Typically, this function is used for milling.
Feedrate adaptations for turning work are programmed with G95 and G96.

\section*{Programing Example}

Workpiece with reference radius \(\mathrm{R}=10 \mathrm{~mm}\)
```

N05 G00 C0
N10 G01 C180 F1000 (Rotational speed of the workpiece 1000 %/min)
(Feedrate at workpiece circumference 174.67 mm/min)
N20 \#FGROUP ROT[AX=C REF=10]
N30 G01 C360 F1000 (Feedrate at workpiece circumference 1000 mm/min)
(Rotational speed of the workpiece 5727.6 o/min)
:
NxX \#FGROUP ROT (Deselection)

```
:
N10 G00 X0 YO ZO
N15 \#FGROUP ROT[AX=C REF=10] (Feedrate at milling cutter contact point \(1000 \mathrm{~mm} / \mathrm{min}\) )

N60 G01 G91 X10 C57.325 F1000 (Diagonal on lateral surface)
N70 G90 X0 C0
Nxx \#FGROUP ROT (Deselection)
:

Independent of the default setting in the channel parameters, the command \#FGROUP WAXIS defines that the axis with the longest running time ("weakest axis") is moved automatically as the feed axis at the programmed feedrate ( F word). All other axes are treated as tracking axes.
\#FGROUP WAXIS (modal)

\section*{Programing Example}
```

N10 \#FGROUP [X, Y] (X and Y are feed axes)
N20 G00 X0 Y0
N30 \#FGROUP WAXIS (Weakest axis is feed axis)
N40 G01 F1000 X10 Y200 (Y axis is axis with longest motion time)
; ..
N999 M30

```

\subsection*{12.21 Adapt path dynamic limit values (\#VECTOR LIMIT ON/OFF)}

\section*{Release Note}

As of Build V2.10.1507.02 the command \#VECTOR LIMIT ON/OFF... replaces the commands \#VECTORACC ON/OFF... and \#VECTORVEL ON/OFF... For compatibility reasons, the commands continue to be available but they should not be used in new NC programs.

The maximum permissible velocity, acceleration and deceleration on the path depend on the dynamic characteristics set in the axis-specific parameter lists and the programmed contour.

To ensure the best results in specific applications (e.g. high-intensity laser or plasma torch cutting processes), it should be possible to modify and adapt the dynamic characteristics on the path directly in NC program.

These path limit values can be adapted by the following NC commands during the dynamic phases of machining. They permit the activation/deactivation of self-defined limits and default limits in the NC program.

It is also possible to limit radial acceleration and radial jerk that occur in curved contour elements (polynomials, circles).

Adapting dynamic path limit values:
```

\#VECTOR LIMIT ON [ [ ACC=.. ] [ DEC=.. ] [ RADIAL_ACC=.. ] [ RADIAL_JERK=.. ] [ JERK=.. ]
[ TRANS_ACC=..][ VEL=.. ] [ FEED ] [ RAPID ] {\}] (modal)
or
\#VECTOR LIMIT ON [ [ ACC ] [ DEC ] [ RADIAL_ACC ] [ RADIAL_JERK ] [ JERK ]
[ TRANS_ACC ] [ VEL ] [ FEED ] [ RAPID ] { \ } ]
(modal)
or
\#VECTOR LIMIT ON ALL (modal)

```
\begin{tabular}{ll}
\(\mathrm{ACC}=<a c c>\) & Path acceleration limit in \(\left[\mathrm{mm} / \mathrm{min}^{2}\right.\), inch \(\left./ \mathrm{min}^{2}\right]\) (default). The unit can be switched by P- \\
CHAN-00351 to \(\left[\mathrm{mm} / \mathrm{sec}^{2}, \mathrm{inch} / \mathrm{sec}^{2}\right]\). \\
\(\mathrm{DEC}=<d e c>\) & Path deceleration limit in \(\left[\mathrm{mm} / \mathrm{min}^{2}\right.\), inch \(\left./ \mathrm{min}^{2}\right]\) (default). The unit can be switched by P- \\
& CHAN-00351 to \(\left[\mathrm{mm} / \mathrm{sec}^{2}, \mathrm{inch} / \mathrm{sec}^{2}\right]\).
\end{tabular}

\section*{Notice}

DEC is effective if the selected slope profile allows a separate definition of acceleration and deceleration parameters (e.g. slope type TRAPEZ [ 372]).

RADIAL_ACC=<r_acc>

RADIAL_JERK=<r_jerk>

VEL=<vel>

FEED
RAPID

1

JERK=<jerk> Limit for path jerk. The unit is always in \(\left[\mathrm{mm} / \mathrm{sec}^{3}\right.\), inch \(\left./ \mathrm{sec}^{3}\right]\) and cannot be switched over. The programming for this parameter is available as of Build V3.1.3079.12.
TRANS_ACC=<tr_acc> Path acceleration limit in block transition in [mm/min \({ }^{2}\), inch \(/ \mathrm{min}^{2}\) ] (default). The unit can be switched by P-CHAN-00351 to [mm/sec\({ }^{2}\), inch/sec\(\left.{ }^{2}\right]\). The programming for this parameter is available as of Build V3.1.3072.02.
Radial acceleration limit in [mm/min \({ }^{2}\), inch \(/ \mathrm{min}^{2}\) ] (default). The unit can be switched by P-CHAN-00351 to [mm/sec \({ }^{2}\), inch/sec \({ }^{2}\) ]. The programming for this parameter is available as of Build V2.11.2033.05.
Limit for radial jerk in [mm/min², inch/min²] (default). The unit can be switched by P-CHAN-00351 to [mm/sec \({ }^{2}\), inch/sec \({ }^{2}\) ]. The programming for this parameter is available as of Build V3.1.3076.02.

Velocity limit [mm/min, inch/min].
If no limit values are programmed for the keywords ACC, DEC, RADIAL_ACC and VEL or the command \#VECTOR LIMIT ON ALL is used, the default dynamic values from the channel parameter list are used (P-CHAN-00002, P-CHAN-00208, P-CHAN-00361, P-CHAN-00090). An internal limit value is set for RADIAL_JERK, JERK and TRANS_ACC.

The command \#VECTOR LIMIT OFF... causes a switch-over to the calculation of the dynamic limitation of the look ahead function. Switching can be programmed both for specific limits and for all limits.
Dynamic limitations are only effective for feedrate blocks (G01, G02, G03).
Dynamic limitations are only effective for rapid traverse blocks (G00).
If the keywords FEED and RAPID are not programmed (default) or both keywords are programmed in combination, the dynamic limitations are effective for all motion blocks (G00, G01, G02, G03).
Separator ("backslash") for clear programming of the command over multiple lines.

Adapting dynamic path limit values:
```

\#VECTOR LIMIT OFF [ [ ACC ][ DEC ] [ RADIAL_ACC ] [ RADIAL_JERK ] [ JERK ]
[TRANS_ACC][VEL]{\}]
(modal)

```
or
\#VECTOR LIMIT OFF ALL

\section*{Notice}

The path dynamic limits are only used if they are smaller than the valid look-ahead limits. They have no influence on axis-specific motions such as homing, manual mode or independent axes and they act on both G01 and G00.

\section*{Attention}

In connection with the channel parameter P-CHAN-00097 the user must take into consideration that machine deceleration during feedhold is also slower depending on the programmed limit.

\section*{Notice}

The path dynamic limits for G00 can also be influenced by a motion path depending on the weighting table defined in the channel parameters [1] [ \({ }^{\text {8 819]-7. }}\)

\section*{Programing Example}

\section*{Adapting path dynamic limit values}
```

%vec_limit
(Set dynamic limit data to specific values)
N10 \#VECTOR LIMIT ON [ACC=3600000 DEC=4000000 VEL=3000 FEED]
N11 \#VECTOR LIMIT ON [ACC=3600000 DEC=4000000 TRANS ACC=3000000]
N12 \#VECTOR LIMIT ON [ACC=3600000 VEL=3000 RADIAL_ACC=2000000 RAPID]
N13 \#VECTOR LIMIT ON [ACC=3600000]
N14 \#VECTOR LIMIT ON [DEC=4000000 VEL=3000 FEED RAPID]
N15 \#VECTOR LIMIT ON [DEC=4000000]
N16 \#VECTOR LIMIT ON [VEL=3000]

```
(Set dynamic limit data to default values)
N20 \#VECTOR LIMIT ON [ACC DEC RADIAL_ACC TRANS_ACC VEL RAPID]
N21 \#VECTOR LIMIT ON [ACC DEC RADIAL_JERK JERK]
N22 \#VECTOR LIMIT ON [ACC VEL FEED]
N23 \#VECTOR LIMIT ON [ACC]
N24 \#VECTOR LIMIT ON [DEC VEL FEED RAPID]
N25 \#VECTOR LIMIT ON [DEC]
N26 \#VECTOR LIMIT ON [VEL]
(Mixed assignment of dynamic limit data)
N27 \#VECTOR LIMIT ON [ACC=3600000 DEC]
N28 \#VECTOR LIMIT ON [ACC VEL=3000]
(Set all dynamic limit data to default values)
N30 \#VECTOR LIMIT ON ALL
(:= \#VECTOR LIMIT ON [ACC DEC RADIAL_ACC RADIAL_JERK JERK TRANS_ACC VEL FEED RAPID])
(Set dynamic limit data by LOOK AHEAD)
N40 \#VECTOR LIMIT OFF [ACC DEC VEL]
N41 \#VECTOR LIMIT OFF [ACC DEC]
N42 \#VECTOR LIMIT OFF [ACC VEL]
N43 \#VECTOR LIMIT OFF [ACC]
N44 \#VECTOR LIMIT OFF [DEC VEL]
N45 \#VECTOR LIMIT OFF [DEC]
N46 \#VECTOR LIMIT OFF [VEL]
(Set all dynamic limit data by LOOK_AHEAD)
N50 \#VECTOR LIMIT OFF ALL
(:= \#VECTOR LIMIT OFF [ACC DEC RADIAL ACC TRANS ACC VEL])
N999 M30

\subsection*{12.22 Defining a minimum block transition velocity (\#TRANSVELMIN ON/OFF)}

At discontinuous block transitions (contour knee angles), path velocity is reduced to such an extent that the resulting axis accelerations do not exceed the limit values predefined in the axis parameters. From a technological aspect (e.g. flame or gas cutting), this velocity reduction at contour knee angles sometimes is undesirable. In this case, the following command defines a minimum velocity which may not be undershot at contour knee angles.

\section*{Attention}

The programmed minimum velocity in the command acts only at discontinuous block transitions of circular and linear blocks. As a result, no contouring mode or spline interpolation may be selected in the NC program (e.g. G261, \#HSC, G151).
```

\#TRANSVELMIN ON [ [ <vel> ] ]
(modal)
\#TRANSVELMIN OFF

```
<vel> Minimum block transition velocity in [mm/min, inch/min].

If no limit value is programmed after \#TRANSVELMIN ON, the minimum transition velocity is set to 0 . The command \#TRANSVELMIN OFF causes a switch-over to the free calculation of the velocity limitation by the look ahead function.

The function works with linear and non-linear slopes but with a non-linear slope, jerk limitation for discontinuous block transitions with channel parameter P-CHAN-00009 (corr_v_trans_jerk=1) may not be active. In such cases, jerk limitation is prioritised.
The programmed minimum velocity is only valid until program end. It is set to zero at the next program start or RESET.
Pre-assignment in the channel parameter list is not possible.

\subsection*{12.23 Writing machine data (\#MACHINE DATA)}


\section*{Release Note}

The availability of this function depends on the configuration and on the version scope.

This command permits a change in axis-specific machine parameters in the NC program. The new values are valid program global. They are overwritten by the next update of machine data lists (e.g. at controller start-up).

\section*{Notice}

An active path interpolation is stopped until the new parameter is adopted and the value is effective. Any rotating spindles are not stopped.
\#MACHINE DATA [<mode>] [ AX<Name> | AXNR<expr> <Param_ID><expr> |
<Param_ID>\{.<idx>\}<expr>| AXPARAM "<string>" [WAIT ]] (modal)
<mode> Synchronisation mode
\begin{tabular}{|c|l|}
\hline Valid modes & Meaning \\
\hline--- & Synchronisation at decoding level (initial state) \\
\hline SYN & Synchronisation at interpolation (real time) level \\
\hline
\end{tabular}
\begin{tabular}{ll} 
AX<Name> & \begin{tabular}{l} 
Name of the related axis or spindle where a new axis-specific parameter is to be writ- \\
ten
\end{tabular} \\
AXNR<expr> & Logical axis number of the path axis or spindle where a new axis-specific parameter is \\
to be written. Positive integer.
\end{tabular}
<Param_ID><expr> Axis parameter in ISG notation (P-AXIS-xxxxx) with new value in the unit of the axis parameter list [AXIS].
<Param_ID>\{.<idx>\}<expr> With axis parameters that address an array, the corresponding element is written by the extended specification of point and index (e.g. P-AXIS-00209.0).

\section*{Notice}

A slave axis in a hard gantry combination can only be addressed by the logical axis number.

The following axis parameters are available via predefined keywords (Param_ID):
\begin{tabular}{|l|l|}
\hline Param_ID & Meaning \\
\hline P-AXIS-00001 & \begin{tabular}{l} 
Non-linear velocity profile: \\
Axis acceleration at velocity increase
\end{tabular} \\
\hline P-AXIS-00002 & \begin{tabular}{l} 
Non-linear velocity profile: \\
Axis acceleration at velocity decrease
\end{tabular} \\
\hline P-AXIS-00004 & Acceleration in rapid traverse (G00) \\
\hline P-AXIS-00005 & \begin{tabular}{l} 
Linear velocity profile: \\
Acceleration step 1 in rapid traverse (G00)
\end{tabular} \\
\hline P-AXIS-00006 & \begin{tabular}{l} 
Linear velocity profile: \\
Acceleration step 2 in rapid traverse (G00)
\end{tabular} \\
\hline P-AXIS-00008 & Permissible axis dynamics: Maximum permissible axis acceleration \\
\hline P-AXIS-00011 & \begin{tabular}{l} 
Linear velocity profile: \\
Acceleration step 1 in feed interpolation (G01, G02, G03)
\end{tabular} \\
\hline P-AXIS-00012 & \begin{tabular}{l} 
Linear velocity profile: \\
Acceleration step 2 in feed interpolation (G01, G02, G03)
\end{tabular} \\
\hline P-AXIS-00045 & Minimum distance (safety distance) between two collision axes \\
\hline P-AXIS-00056 & Maximum difference after deselecting tracking mode \\
\hline P-AXIS-00075 & \begin{tabular}{l} 
Gantry mode:Velocity for compensating for static offset between mas- \\
ter and slave axes due to differences in measuring systems
\end{tabular} \\
\hline P-AXIS-00099 & Acceleration factor kv for P positional control \\
\hline P-AXIS-00103 & Size of backlash \\
\hline P-AXIS-00109 & \begin{tabular}{l} 
Maximum permissible velocity override for independent axis and \\
spindles
\end{tabular} \\
\hline P-AXIS-00151 & Maximum transient time to reach the exact stop window \\
\hline P-AXIS-00152 & Absolute position of reference point \\
\hline P-AXIS-00166 & Remaining deviation for non-linear position lag monitoring \\
\hline P-AXIS-00167 & Factor for parameterising dynamic position lag monitoring \\
\hline P-AXIS-00168 & Maximum position lag \\
\hline P-AXIS-00169 & Minimum position lag \\
\hline P-AXIS-00172 & Type of position lag monitoring \\
\hline P-AXIS-00195 & Non-linear velocity profile: Ramp time for acceleration down-gradation \\
\hline P-AXIS-00196 & Non-linear velocity profile: Ramp time for acceleration up-gradation \\
\hline P-AXIS-00197 & Non-linear velocity profile: Ramp time for deceleration down-gradation \\
\hline & \\
\hline
\end{tabular}
\begin{tabular}{|l|l|}
\hline P-AXIS-00198 & Non-linear velocity profile: Ramp time for deceleration up-gradation \\
\hline P-AXIS-00200 & Non-linear velocity profile: Ramp time in rapid traverse (G00) \\
\hline P-AXIS-00201 & Minimum permissible ramp time of the drive to limit axis jerk \\
\hline P-AXIS-00208 & \begin{tabular}{l} 
Maximum feed of compensation motion after deselecting tracking op- \\
eration
\end{tabular} \\
\hline P-AXIS-00209 & Rapid traverse velocity G00 \\
\hline P-AXIS-00211 & \begin{tabular}{l} 
Linear velocity profile: \\
Changeover speed in rapid traverse (G00) between velocity ramp 1 \\
and velocity ramp 2
\end{tabular} \\
\hline P-AXIS-00212 & Permissible axis dynamics: Maximum permissible axis velocity \\
\hline P-AXIS-00216 & \begin{tabular}{l} 
Minimum permissible axis velocity for spindles. Below this velocity, \\
the rotational speed monitor in the position controller supplies the \\
"speed zero" state
\end{tabular} \\
\hline P-AXIS-00217 & \begin{tabular}{l} 
Factor to calculate the actual rotational speed at which the "speed \\
setpoint reached" state is signalled.
\end{tabular} \\
\hline P-AXIS-00218 & Minimum homing velocity \\
\hline P-AXIS-00219 & Maximum homing velocity \\
\hline P-AXIS-00221 & \begin{tabular}{l} 
Linear velocity profile: \\
Changeover speed in feed interpolation (G01, G02, G03) between ve- \\
locity ramp 1 and velocity ramp 2
\end{tabular} \\
\hline P-AXIS-00236 & Size of control window for exact stop \\
\hline P-AXIS-00414 & Maximum position offset for distance control \\
\hline
\end{tabular}

AXPARAM "<string>"

Alternative syntax: Axis parameter with complete structure path and value in internal notation of the axis parameter list [AXIS] (see example). It is possible to write all axis parameters.
This keyword may only be used in conjunction with the synchronised setting (SYN) of an axis parameter. If a WAIT is programmed, program decoding is also interrupted (implicit FLUSH) and the program waits until the new parameter is adopted and becomes effective in the NC channel.

\section*{Programing Example}

Writing machine data

kernel Industrielle Steuerungstechnik GmbH

\subsection*{12.24 File operations}

\subsection*{12.24.1 Definition of file names (\#FILE NAME)}

The command \#FILE NAME defines file names in NC program file. The file names are used by specific NC functions, e.g. to create a report file or to call a NC program.
```

\#FILE NAME [ <File_ID>="<filename>" { <File_ID>="<filename>" }] (modal)

```
<File_ID> File identification
\begin{tabular}{|l|l|}
\hline File id's & Meaning \\
\hline MSG & \begin{tabular}{l} 
Name of the report file for \#MSG SAVE. \\
Default name is "message.txt".
\end{tabular} \\
\hline M6 & \begin{tabular}{l} 
Name of the global subroutine for implicit program call via M6 in the NC pro- \\
gram. The name is valid until M30. M6 is not treated as an M function any \\
more. \\
The default name is set in P-CHAN-00118. M6 is always executed as the \\
last action at block end.
\end{tabular} \\
\hline D & \begin{tabular}{l} 
Name of the global subroutine for implicit program call by a D word in the \\
NC program. The name is valid until M30. \\
The default name is set in P-CHAN-00429. The subroutine call is always ex- \\
ecuted as the last action at block end.
\end{tabular} \\
\hline G80 - G89 & \begin{tabular}{l} 
Name of the global subroutines for implicit program calls at G80 -G89 in the \\
NC program. The names are valid until M30. \\
Default names are set in P-CHAN-00160 - P-CHAN-00169. G80-G89 is al- \\
ways executed as the last action at block end.
\end{tabular} \\
\hline \begin{tabular}{l} 
G800- G819 \\
or \\
G800- G839**
\end{tabular} & \begin{tabular}{l} 
Names of the global subroutines in case of additional implicit program calls \\
at G800 - G819*** in the NC program. The names are valid until M30. \\
Default names are set in P-CHAN-00187. A G800-G819** is always ex- \\
ecuted as the last action at block end. \\
** Extended to 40 calls (G800 - G839) as of Build V3.1.3079.23
\end{tabular} \\
\hline
\end{tabular}

The equals sign (=) is optional.
The file names can be defined or changed in the NC program with \#FILE NAME at any time.
File names are set to their default names at RESET and at NC program start.

\section*{Programing Example}
```

%example1
N10 \#FILE NAME[ MSG="prog_flow.txt" ]
N20 \$IF V.E.PLC_START_HOME == 1
N30 G74 X1 Y2 Z3
N40 \#MSG SAVE["Homing executed"] ;Output in prog_flow.txt
N50 \$ENDIF
;..
N120 \#MSG SAVE["Roughing OK"]
; ...
N950 \#MSG SAVE["Finishing OK"]
N985 V.E.WP CNTR = V.E.WP CNTR+1
N990 \#MSG SA\overline{VE["Workpiece No. %d OK", V.E.WP_CNTR]}
M1000 M30

```
\%example2
N10 \#FILE NAME [ M6="tool_change.nc" ]
N20 G00 X100 Y100 Z0
N30 M6 ; Call tool change program tool_change.nc
; ...
M1000 M30
\%example3
N10 \#FILE NAME[ D="tool_data.nc" ]
N20 G00 X100 Y100 Z50
N30 D1 ;1:Request data of tool 1
    ;2:Implicit call of tool_data.nc
;...
M1000 M30
\%example4
N10 \#FILE NAME [ G80="g80_up_test.nc" ]
N20 G00 X100 Y100 Z50
N30 G80 ; Call subroutine g80_up_test.nc
; ...
M1000 M30
```

%example5
N10 \#FILE NAME[ G800="g800_up_test.nc" G815="g815_up_test.nc"]
N20 G00 X100 Y100 Z50
N30 G800 ; Call subroutine g800_up_test.nc
; ...
N90 G815 ; Call subroutine g815_up_test.nc
;..
M1000 M30

```

\subsection*{12.24.2 Renaming a file (\#FILE RENAME)}

The command \#FILE RENAME renames an existing file. All parameters must be specified. Omitting a parameter leads to a corresponding error message.
\#FILE RENAME [ PATHOLD="<filename>" PATHNEW="<filename>" OVRMODE=<expr> ]

PATHOLD<filename> File to be renamed with directory specification. If this directory path or the file does not exist, the NC program is aborted with an error message.
PATHNEW<filename> New (destination) file with directory specification. If a different directory path than the default directory is specified here, the file is moved to the directory with the new name, provided the directory exists. If this directory is not found, a corresponding error message is output.
OVRMODE<expr>
Boolean value indicating whether a file specified by PATHNEW should be overwritten provided it already exists.
0 : File may not be overwritten. output of an error message
1: Existing file may be overwritten.

The equals sign (=) is optional.

\section*{Notice}

The user must have write access to the directories PATHNEW and PATHOLD. Otherwise, renaming causes an error.


\section*{Attention}

WRITE PROTECTION:
An error is generated if the file which is to be renamed is write-protected, it is an existing (destination) file and is protected.


\section*{Attention}

RELATIVE DIRECTORIES:
If PATHOLD is specified as relative, a search is made for the file in the folders of the start-up/ channel parameter list.
The search is for the sequence main program - subroutine - work directory. If PATHNEW is specified as relative, the PATHOLD directory is used.

\section*{Notice}

In addition, the default directory for the file operation is used in TwinCAT. This setting is made in the System Manager.

\section*{Programing Example}

Rename a file
\%FileRename

N10 \#FILE NAME[MSG="C:\Test.txt"] ; Create file
N40 \#MSG SAVE["Write me into file"] ;Writes text to file

N60 \#FILE RENAME[PATHOLD="C:\Test.txt" PATHNEW = "C:\NewName.txt" OVRMODE=1]
N70 M30

\subsection*{12.24.3 Deleting a file (\#FILE DELETE)}

The command \#FILE DELETE is used to delete a file. The parameter must be specified. Otherwise, a corresponding error message is generated.
```

\#FILE DELETE [ PATH="<filename>" ]

```

PATH<filename> File to be deleted with directory specification.

The equals sign (=) is optional.

\section*{Notice}

The user must have write authorisation to the PATH Directory to be able to delete a file.
\(\qquad\) Attention
WRITE PROTECTION:
If the file is write-protect5ed, the error is generated by ID 21627.
\(\square\) \begin{tabular}{l} 
Attention \\
RELATIVE DIRECTORIES: \\
If the PATH parameter is specified as relative, a search is made for the file in the folders of the \\
start-up/channel parameter list. \\
The search is for the sequence main program - subroutine - work directory. \\
\hline Notice \\
In addition, the default directory for the file operation is used in TwinCAT. This set- \\
ting is made in the System Manager.
\end{tabular}

\section*{Programing Example}
```

%FileDelete
N10 \#FILE NAME[MSG="C:\Test.txt"] ;Create file
N40 \#MSG SAVE["Write me into file"] ;Writes text to file
N60 \#FILE DELETE[PATH="C:\Test.txt" ] ; Delete file
N70 M30

```

\subsection*{12.24.4 Checking existence of a file (\#FILE EXIST)}

The command \#FILE EXIST checks whether a file exists in the file system and can open it. After calling \#FILE EXIST the call result is saved to V.G. variables. This is retained until the next time \#FILE EXIST is called.
\begin{tabular}{|c|c|c|c|c|}
\hline V.G.<var_name> & Meaning & Data type & Unit of In/Output & Permitted access Read/ Write \\
\hline FILE_EXIST & Result of data search. When the file is found, then 1 & Boolean & 0,1 & L \\
\hline FILE_EXIST_PATH_NAME * & Path with file name* & String & - & L \\
\hline FILE_EXIST_DIRECTORY & Path without file name. The path ends with 'l' & String & - & L \\
\hline
\end{tabular}
* Paths that are entered in the table of search paths in the start-up list (P-STUP-00018) or channel list (P-CHAN-00401).
```

\#FILE EXIST [ PATH="<filename>" ]

```

PATH<filename> Name of the file whose existence is to be checked with or without directory specification.

The equals sign (=) is optional.

Attention
RELATIVE DIRECTORIES:
If the PATH parameter is specified as relative, a search is made for the file in the folders of the start-up/channel parameter list.

The search is for the sequence main program - subroutine - work directory.

\section*{Programing Example}

Check for the existence of a file
```

%FileExist
NO10 \#FILE EXIST[PATH = "C:\TestDir\test.nc"]
N030 \$IF V.G.FILE_EXIST == TRUE
NO40 \#MSG ["FILE EXISTS"]
N050 \$ELSE
N060 \#MSG ["FILE DOES NOT EXIST"]
NO70 \$ENDIF
N090 M30

```

\subsection*{12.24.5 Create and manage backup files}

\section*{Logging data}

The user can write text to file in the NC program using \#MSG SAVE [...]. New data is appended to the file, causing the file size to gradually increase.

\section*{Defining size and number of log files}

To check the size and access to large files, the user can limit the maximum number of lines in the file. When the maximum size is reached, the file is backed up automatically and the original file is deleted.

To track the chronology of file changes, several backups (log files) can be saved. The user can also define the maximum number of backups. When the maximum number of backup files is reached, the oldest back file is discarded.

The size and number of backup files is defined with the command \#FILE NAME [...]. This refers to the file which is written with \#MSG SAVE.
\#FILE NAME [ BACKUP_LINES_MAX=<Lines> BACKUP_COUNT_MAX=<Number> ] (modal)

BACKUP_LINES_MAX<Lines>
BACKUP_COUNT_MAX<number>

Number of lines in the file since controller start-up. If this is overshot, a backup is created. This applies to the files last programmed by <number>. Maximum number of backups saved in parallel. If the maximum number is reached when a new backup is created, the older backup is overwritten.


Fig. 112: Diagram of backup function


Fig. 113: Creating backup files

\section*{Activating and deactivating the backup function}

The backup function is activated by specifying a maximum number of lines \(>0\) and a maximum number of files \(>0\).

Deactivation takes place if one of these parameters is programmed with 0 or none of these parameters is programmed or at program end M30.

\section*{Notice}

The current maximum number of lines corresponds to the number of write accesses per \#MSG SAVE command. In other words, the actual number of lines is not determined. This produces a difference in particular if a message containing several lines is written.

The original name of the backup file is specified by \#FILE NAME [MSG=<name>]. The name of the backup consists of the original name, the count number <no> and the file extension .bak.

Example: Test.log (original file)
Test1.bak, Test2.bak, Test3.bak, etc.

\section*{Behaviour at controller restart}

When the controller is started, back files may already exist. The first time a new backup is created, a search is made for the oldest backup. This means that after controller start, a new backup always starts with 1.

\section*{Notice}

A backup is only deleted if it is necessary when a new backup is created.
No backup is automatically deleted at controller start-up, when the backup behaviour is defined by the NC command, in the event of an error or at NC reset.

The name of the last backup file and the current lines are saved. This means that the current state is always updated when the backup function is re-activated. This data is only deleted at controller restart. All other values for previously programmed filenames are discarded.

\subsection*{12.25 Monitoring the work space and protection space}

The use of control areas prevents specific axis configurations from causing collisions between the tool and parts to be protected. A control area may be a work space or a protection space.

Control areas can be defined as three dimensional objects in cylindrical or polygonal form. The third dimension is defined by constant minimum and maximum values. Axes are assigned dependent on the currently active working plane (e.g. G17).


Fig. 114: Definition of 3D control areas (cylindrical, polygonal)


Fig. 115: Example of cylindrical workspace areas in an application

It is not possible to move out of a work space and into a protection space. Current tool data are included and the tool tip can be checked for validity within the control area.
Work and protection spaces are defined, activated and de-activated in the NC program. Several control areas may be active at the same time. Control areas can be redefined in de-activated state.

Work space and protection space monitoring with tool centre point monitoring is possible with
- automatic mode in conjunction with:
- Linear motion blocks
- Circular motion blocks (irrespective of their orientation G17/G18/ G19).
- Kinematic transformations
- Polynomial contours (contouring, HSC)
- Helical motions
- Reference point offsets with G92, G54
- Cartesian transformations \#(A)CS available as of CNC Build V2.11.2015:
- active manual mode available as of CNC Build V 3.1 .3068 .9 : in conjunction with:
- Exclusive (G200) or inclusive mode (G201/G202)
- Kinematic transformations

\subsection*{12.25.1 Defining a control area (\#CONTROL AREA BEGIN/END)}

\section*{Time of definition}

No workspace/protection areas are predefined when the controller starts up. A definition in the configuration lists is not possible.
A work or protection space is defined directly in the NC program in a sequence of path motions embedded in plain text commands.
In this case, path motions must always be programmed in absolute dimensions. The contour of the control area in the plane is defined either by a closed polygon formed by linear blocks (target point and starting point of the block sequence must be identical) or by a full circle. The excursion in the third dimension and further characteristics of the control area are defined in the assigned plaintext command.

\section*{Start of control area definition:}
```

\#CONTROL AREA BEGIN [ ID<expr> WORK | PROT POLY | CIRC
MIN_EXCUR<expr> MAX_EXCUR<expr>
[EXCUR_AX<expr>|EXCUR_AXNR<expr>]]
(modal, program global and CNC reset global)

```
\begin{tabular}{ll} 
ID<expr> & \begin{tabular}{l} 
Identification number of the control area (ID). The definition is global valid after pro- \\
gram end and RESET. Up to 20 different control areas can be defined. \\
Control area is a workspace.
\end{tabular} \\
WORK & \begin{tabular}{l} 
Control area is a protection space. \\
PROT
\end{tabular} \\
Contour of a control area is defined as a polygonal shape. \\
POLY & Contour of a control area is defined as a full circle. \\
CIRC & Limitation of the control area in the third dimension in negative direction in [mm, inch]. \\
MIN_EXCUR<expr> & \begin{tabular}{l} 
Limitation of the control area in the third dimension in positive direction in [mm, inch]. \\
MAX_EXCUR<expr> \\
EXCUR_AX<expr>
\end{tabular} \\
\begin{tabular}{l} 
Optional specification of an axis identifier for the third excursion direction of the work- \\
space or protection area (as of CNC Build V2.11.2025.00). By default the third main \\
axis is used.
\end{tabular} \\
EXCUR_AX<expr> & \begin{tabular}{l} 
Optional specification of a logical axis number for the third excursion direction of the \\
workspace or protection area (as of CNC Build V2.11.2025.00). By default the third \\
main axis is used.
\end{tabular}
\end{tabular}

\section*{End of control area definition:}
\#CONTROL AREA END (modal)

Each control area must be closed by the command \#CONTROL AREA END. Only then can further control areas be defined.


\section*{Attention}

Active Cartesian transformations \#(A)CS are not considered in the definition of monitored spaces. Workspaces and protection areas are always defined as Cartesian in the MCS coordinate system.

\section*{Programing Example}

Define control areas
```

(Define a polygonal workspace:
:
N10 \#CONTROL AREA BEGIN [ID1 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X-150 Y75 (Starting point)
N30 X-50 Y150
N40 X50 Y150
N50 X150 Y75
N60 X150 Y0

```
```

N70 X50 Y0
N80 X50 Y75
N90 X-50 Y75
N100 X-50 Y0
N120 X-150 Y0
N130 X-150 Y75 (End point identical to starting point)
N140 \#CONTROL AREA END
:
(Define a cylindrical protection area:
:
N210 \#CONTROL AREA BEGIN [ID2 PROT CIRC MIN_EXCUR=-70 MAX_EXCUR=70]
N220 G01 X100 Y0 F10000 (Starting point for cylindrical prot. area)
N230 G02 G162 I50 J0
N240 \#CONTROL AREA END

```
:


\subsection*{12.25.2 Selecting/deselecting control areas (\#CONTROL AREA ON/OFF)}

When control areas are activated the TCP must already be in the valid work space. In the same way, when a work space is activated, the TCP may not cause a violation at the current position.
\#CONTROL AREA ON [ALL] | [ ID<expr> ] (modal)

ID<expr> Identification number of the control area (ID).
ALL Activate all defined control areas.

\section*{Programing Example}

Deselecting and selecting control areas
```

\#CONTROL AREA ON ALL (Activate all defined control areas)

```
\#CONTROL AREA ON [ID3] (Activate specific control area)
\#CONTROL AREA OFF [ALL] | [ ID<expr> ] (modal)

ID<expr> Identification number of the control area (ID).
ALL
Deactivate all currently defined control areas.

\section*{Programing Example}
```

\#CONTROL AREA OFF [ID3] (Deactivate specific control area)
\#CONTROL AREA OFF
\#CONTROL AREA OFF ALI

### 12.25.3 Clearing control areas (\#CONTROL AREA CLEAR)

\#CONTROL AREA CLEAR [ALL]|[ ID<expr>] (modal)
ID<expr> $\quad$ Identification number of the control area (ID).

Delete all defined control areas.

## Programing Example

Deleting control areas

```
#CONTROL AREA CLEAR [ID3] (Delete specific control area)
#CONTROL AREA CLEAR ALL (Delete all defined control areas)
```

kernel Industrielle Steuerungstechnik GmbH

### 12.25.4 Monitor additional axes

Besides the main axes. $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ can adopt additional axes in the monitoring function for work spaces and protection spaces. In this case, the definition of the assigned control areas is limited to polygonal shapes. The control areas are defined using the associated axis identifiers.

## Programing Example

Define a work space for the additional axes X2, Y2 and Z2

```
N10 #CONTROL AREA BEGIN [ID4 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X2=100 Y2=100 ; Starting point
N30 X2=-100
N40 Y2=-100
N50 X2=100
N60 X2 = 100 Y2= 100 ; End point identical with starting point
N70 #CONTROL AREA END
:
N500 #CONTROL AREA ON ALL
:
N1000 M30
```


### 12.25.5 Special features in manual mode

## Exclusive manual mode (G200)

In response to an error in automatic mode, an error message is output and the program is aborted. However in manual mode, only an axis motion stop occurs as an error reaction when a person enters the protection space or leaves the workspace. This response can be compared to approaching manual mode offset limits.

## Inclusive manual mode (G201/G202)

If motions from automatic and manual mode are superimposed (parallel interpolation), work space and protective space violations may occur.

## Output a warning message

The reason for motion stop is displayed to the user by the output of a warning message. To achieve this, P-MANU-00014 must be set.

## Suppress workspace monitoring

The parameter P-CHAN-00442 influences or even suppresses workspace monitoring in manual mode.

## Example

Suppress workspace monitoring with P-CHAN-00442

Initial situation: The machine runs in automatic or manual mode. Before activating manual mode, activate workspace monitoring, e.g. In a subroutine.

## Case 1:

P-CHAN-00442 is assigned the value 1.
Workspace monitoring is not activated in manual mode although its definition and activation were executed in the NC program. The machine can move back and forth across workspace boundaries.

## Case 2:

P-CHAN-00442 is assigned the value 0.
Workspace monitoring is active in manual mode in combination with the Suppress error output from workspace monitoring in manual mode control unit.

When manual mode is active, workspace monitoring can be deactivated by the signal set in the Suppress error output from workspace monitoring in manual mode control unit. The machine can then move back and forth across workspace boundaries.

When manual mode is activated the TCP must be within the permitted range. If this is not the case, an error is output.
Error ID 50961, if the workspace was left
Error ID 50962, if the protection space was left.

Case 3:
P-CHAN-00442 is assigned the value 2.
Workspace monitoring is activated in manual mode in combination with the Suppress error output from workspace monitoring in manual mode control unit.

When manual mode is activated, the TCP may be located outside the permitted range. The position of the TCP is not checked, Workspace monitoring can be deactivated by the control unit. The machine can move back within the permitted range.

### 12.26 Influence forward/backward motion on path



## Release Note

The availability of this function depends on the configuration and the scope of the version.

For more information on these commands and forward/backward motion on the path see the functional description [FCT-C7].
12.26.1 Skipping program sequences (\#OPTIONAL EXECUTION)

In the NC program the command \#OPTIONAL EXECUTION flags a sequence of NC blocks as non-executable. The labelled area is then omitted at interpolator level if forward/backward motion is active. It is activated via the PLC with the "simulate_motion" control unit.

No new calculation is performed for the transition conditions between the blocks before or after the omitted area.
\#OPTIONAL EXECUTION [ ON | OFF | CLEAR ]
[ [ SIMULATE ] | [ SIMULATE MASK = .. ] | [ APPROACH] ] (modal)

| ON | Activate skipping |
| :--- | :--- |
| OFF | Deactivate skipping |

This following syntax is available as of CNC Build V3.3107.12.
CLEAR This command delete every movement. This ensures that, in the case of an escape movement, no motion block can be executed by the Escape function before this flag.
SIMULATE The programmed sequence between \#OPTIONAL EXECUTION ON and OFF is only omitted if the signal of the simulate_motion control unit is active.
SIMULATE MASK =<ex- 64-bit mask for the specification.
pr>

APPROACH
The sequence between \#OPTIONAL EXECUTION ON and OFF is only omitted if the signal of the simulate_motion control unit is active and the programmed mask has bitwise matches with the simulate_motion_mask control unit.
Implicitly flags the start and the end of the approach geometry in the Escape channel. This approach geometry is appended to the dynamically generated Escape path in backward direction.

Keyword can only be used for EDM any may only be programmed in the Escape channel.

The interpolator context, especially referring to axis positions, must remain unchanged in order to avoid a discontinuous transition of path, velocity and acceleration between these blocks.

## Notice

Axis positions must be identical before and after the masked sequence.
If axis positions are changed, error ID 50452 is output.

## Synchronisation of M/H functions with \#OPTIONAL EXECUTION [SIMULATE]

By default, M or H functions are output with the synchronisation MOS within a sequence flagged by \#OPTIONAL EXECUTION ON/OFF [SIMULATE].
If the specified synchronisation is used from the channel parameters for the M or H function, the bit
FWD SYNCH 0x800000
must also be set for the associated M or H function.

## Programing Example

Parameterisation M functions with \#OPTIONAL EXECUTION [SIMULATE]

M103 and M104 are synchronised for "simulated" forward motion:

```
m_synch[103] 0x00800002
m_synch[104] 0x00800004
```

Programing Example
Skipping program sequences

```
%t storag.nc
X1\overline{0}}\mathrm{ Y0
N10 G91 G00 X10 F1000
N11 #OPTIONAL EXECUTION ON
N12 Z123
N13 S1000 M3
N14 Z-123
N15 M101
N16 #OPTIONAL EXECUTION OFF
N20 G90 G01 X0
N30 G02 I10
N40 G03 J10
M30
```

The CNC only checks and monitors the continuous path of axes with or without skipping of blocks. Any further conditions must be ensured by the user itself and are not checked by the CNC.

Nesting of multiple commands optional execution on/off is not considered.

If execution is within an OPTIONAL EXECUTION area and a forward/backward motion is selected, skipping is delayed until \#OPTIONAL EXECUTION OFF is reached. This is necessary, because the marked sequence is just allowed to omit at all or not.

Before leaving (M17, M29) the program level where the OPTIONAL EXECUTION was selected (ON), it must also be deselected (OFF). This also applies to leaving the main program level (M30]. If the program level is terminated without deselection, error ID 21719 is output.

Notice
It is impractical to use the NC command \#OPTIONAL EXECUTION together with contour-changing functions, such as tool radius compensation or polynomial smoothing.

### 12.26.2 Clearing backward storage (\#BACKWARD STORAGE CLEAR)

The NC command \#BACKWARD STORAGE CLEAR can explicitly clear the current backward storage. This ensures that afterwards the actually available backward storage cannot be moved further back than the clearing position in the program.
\#BACKWARD STORAGE CLEAR (modal)

## Programing Example

Clearing backward storage
\%backward-storage
N000 G01 X0 F10000
N010 X100 Y123
N020 X100
N030 X200 Y10
NO40 X300 Y20
NO50 \#BACKWARD STORAGE CLEAR
N060 X400 Y-20
N070 X500 Y-3
NO60 \#BACKWARD STORAGE CLEAR

N080 X444 Y10
N090 X333 Y3
N100 X222 Y10
N110 X111 Y3
N120 X000 Y10
N130 X-111 Y3

N140 \#BACKWARD STORAGE CLEAR
N1000 M30

### 12.27 Tool change with active synchronous mode (\#FREE TOOL CHANGE)

For security reasons, it is generally not permitted to adopt new tool data (T (with implicit D), D or \#TOOL DATA) in synchronous mode. This is indicated by error message 20169. Locking can be switched by the following NC command:

## \#FREE TOOL CHANGE ON | OFF <br> (modal)

At the latest, the NC command must be programmed before the first tool change selected in active synchronous mode. The synchronous mode lock is disabled by the keyword "ON".
Locking of synchronous mode and tool change are re-enabled by the keyword "OFF". In addition, the NC command is deselected at RESET and implicitly at the end of the main program.

At program start, locking of synchronous mode and tool change is always active for compatibility and security reasons (default).

## Notice

If the channel parameter P-CHAN-00100 (move_tool_offsets_directly) is active at the same time, the error message 20169 continues to be generated since the immediate exiting of tool offsets in the slave axes could lead to machine damage when synchronous mode is active. This means that when \#FREE TOOL CHANGE is used, P-CHAN-00100 should be deactivated.

## Programing Example

```
%TOOL_AXLINK
N05 X0}\mathrm{ Y0 Z0 C100 G53 D0
N10 #AX LINK [1,C=X]
N15 #FREE TOOL CHANGE ON
N2O #AX LINK ON [1]
N30 X100 Y50 Z30
N40 D2
N50 X200 Y75 Z40
N60 D1
N65 X300 Y100
N70 #AX LINK OFF [1]
N75 #FREE TOOL CHANGE OFF
N70 X25 Y25 Z25 C25
N80 M30
```


### 12.28 <br> Locking program areas for single-step mode (\#SINGLE STEP)

The command \#SINGLE STEP [ DISABLE / ENABLE ] locks any program areas in the NC program for single-step mode. This allows the operator to skip the labelled program areas in one step and jump faster to the NC lines to be analysed.
If locked areas are nested, the single-step lock includes the area starting at the first activation to the first deactivation (see example 2).

The user can continue to define an incremental width at a specific block number single-step resolution \#SINGLE STEP [ RESOLUTION... ] at which the single-step stop is to act (see example 3).
For further information on single-step block mode, see the functional description [FCT-M2].

DISABLE Start of single-step lock.
ENABLE
End of single-step lock.
RESOLUTION<expr>
Specific block number incremental width at which the single-step stop acts.

## Programing Example

Locking program areas for single-step mode
Example 1:

Single-step mode is not effective in the area of the NC blocks N40-N100 including all included subroutines called in it.

```
%SINGLE STEP
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #SINGLE STEP [DISABLE]
N50 X20
N60 Y20
N65 L GSP.nc
N70 Z20
N80 X30
N90 Z30
N100 #SINGLE STEP [ENABLE]
N110 Y30
N120 X40
N130 Z40
N999 M30
```


## Example 2:

Area of single-step lock with nesting includes N40-N75

```
%SINGLE STEP
N10 X0 Y}0 Z
N20 X10
N30 Y10
N40 #SINGLE STEP [DISABLE]
N50 X20
N55 #SINGLE STEP [DISABLE]
N60 Y20
N65 L GSP.nc
N70 Z20
N75 #SINGLE STEP [ENABLE]
N80 X30
N90 Z30
N100 #SINGLE STEP [ENABLE]
N110 Y30
N120 X40
N130 Z40
N999 M30
```


## Example 3:

Block numbering with user-specific single-step resolution (10 steps) and internal numbering (1 step). The black lines show the single-step stop. There is no stopping within the grey area.
\%SINGLE_STEP
N010 \#SINGLE STEP [RESOLUTION = 10]

NO90 Y0
N091 Y1
N092 Y2
N093 Y3
N094 Y4

N100 Y5
N101 Y6
N102 Y7

N110 Y8

### 12.29 Programmable path override (\#OVERRIDE)

This command influences path feed in the NC program if this needs to be different for feed and rapid traverse blocks. The programmed override for path motions is active if at least one axis moves. This does not affect the mode of operation of real-time influencing of feed by the PLC.
An additional function is provided for programmable axis override [ $>790$ ].
If an axis override is also defined for an axis involved in path motion, the effective path override results from multiplication of the two override values.

## Notice

The G166 [» 188] function suppresses the programmed override values.

## \#OVERRIDE [ FEED_FACT<expr> RAPID_FACT<expr> ]

FEED_FACT<expr> Override factor for feed blocks [0.1\%-200\%].
RAPID_FACT<expr> Override factor for rapid traverse blocks [0.1\%-200\%].

## Programing Example

## Programmable path override

```
%path_override
N10 G00 G90 X0 Y0 Z0
    (Path override G01 122.765%, GOO 155.7%)
N20 #OVERRIDE [FEED FACT=122.765 RAPID FACT=155.7]
N30 G01 X100 Y100 Z100 F1000 Effective feed=1227.65
    (Path override G01, GOO 100%)
N40 #OVERRIDE [FEED FACT=100 RAPID FACT=100]
N50 G01 X200 Y200 Z2\overline{200 F1000 Effective feed=1000}
```

M30

### 12.30 Drive-independent switching of drive functions

As opposed to SERCOS-specific commands as described in Section Writing and reading drive parameters and commands [ $>374$ ], the following commands permit the drive-independent setting of drive parameters. The parameters can only be written synchronously, i.e. the command is executed at interpolator level at the time of processing. In the default setting, the \#DRIVE command is executed in parallel to the following processing of the NC program, i.e. program processing is not stopped. The "WAIT" parameter can be used to stop interpolation until the drive function is completed.
The command \#DRIVE WAIT SYN can be used to check the successful switching of the drive function at a later time.

### 12.30.1 Synchronised write (\#DRIVE WR SYN)

\#DRIVE WR SYN [ AX<axis_name> | AXNR<expr> [ MOTOR<expr> ] [PARAM_SET<expr> ]
KEY<string> VAL<expr> [ WAIT ] ]

AX<axis name>
AXNR<expr>
MOTOR<expr>
PARAM_SET<expr>

KEY <string>
VAL<expr>
WAIT

Name of the (drive) axis
Logical axis number of the (drive) axis, positive integer
Selectable motor number in the drive amplifier on motor switching; positive integer
Switch-over parameter record in the drive amplifier;
positive integer
Keyword of addressed function as described in [2] [> 819]
Value to be transmitted, negative or positive integer
Stopping interpolation and waiting for successful switching of drive function.
If this keyword is not specified, a \#DRIVE WAIT SYN can be programmed in the subsequent part of the NC program to check the successful switching of the drive function.

## Attention

When continuous rotation is active for spindles, no motor switching and no changing of the parameter record (MOTOR=<expr>, PARAM_SET=<expr>)<expr>, PARAM_SET=<expr>may be programmed.

### 12.30.2 Synchronous waiting for acknowledgement (\#DRIVE WAIT SYN)

The following command checks whether all previous \#DRIVE WR SYN were completed for an axis. The interpolator is stopped until all \#DRIVE WR SYN are executed in the drive. This applies to both path axes and spindle axes.

```
#DRIVE WAIT SYN [ AX<axis_name> | AXNR<expr> SWITCH_OK ]
```

AX <axis name>
AXNR<expr>
SWITCH_OK

Name of the (drive) axis
Logical axis number of the (drive) axis, positive integer
Check whether all previous \#DRIVE WR SYN are completed.

## Programing Example

Synchronous waiting for acknowledgement

```
Synchronous writing with immediate waiting for acknowledgement:
%TOOL AXLINK1
N05 XO YO ZO
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=torque_limit VAL=400 WAIT]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30
Synchronous writing with subsequent waiting for acknowledgement:
%TOOL_AXLINK2
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=torque_limit VAL=400]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N60 #DRIVE WAIT SYN [AX=X SWITCH_OK]
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30
```


### 12.31 Velocity-optimised motion control by segmentation (\#SEGMENTATION)

In some applications (e.g. kinematics where singular areas can appear) it may be an advantage to improve the programmed block segmentation on the CNC side or to convert circular blocks (G2/ G3) into linear blocks (G1) by segmentation. In addition, circular blocks can also be divided into circle segments to improve the utilisation of machine dynamics. This can be achieved in the NC program by the following command.

```
#SEGMENTATION [ON | OFF] [ALL] [ [ [LIN] [LENGTH<expr>] [CIR] [OPMODE<expr>]
[PARAM<expr>]]]
    (modal)
```

| ON | Select segmentation |
| :--- | :--- |
| OFF | Deselect segmentation |
| ALL | Segment linear and circular blocks |
| LIN | Segment linear blocks |
| LENGTH<expr> | Length of the resulting linear blocks in [mm, inch] <br> CIR |
| Segment circular blocks |  |
| OPMODE<expr> | Mode of circle segmentation: <br> 0: Preset required block length. |
|  | 1: Specify required chordal error. Block length is calculated automatically (default). <br>  <br> 2: Preset required block length, output as circle segments. |
| PARAM<expr> | Chordal error and block length are defined by the keyword PARAM. <br> Either chordal error or length of the resulting linear blocks, depending on the selected <br> OPMODE in [mm, inch] |

If no other parameterisation is programmed except for LIN and/or CIR when segmentation is activated, the following initial state is valid:

| LENGTH | 1 mm |
| :---: | :---: |
| OPMODE | 0 |
| PARAM | 0.1 mm |

## Programing Example

Velocity-optimised motion control by segmentation

Select linear segmentation with default parameterisation:

N20 G00 X0 Y0 Z0 F10000
N30 \#SEGMENTATION ON [LIN]
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 \#SEGMENTATION OFF [LIN] ;Deselect
N100 M30

Select linear segmentation + parameterisation:
N20 G00 X0 Y0 Z0 F10000
N30 \#SEGMENTATION ON [LIN LENGTH 0.5]
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 \#SEGMENTATION OFF [LIN] ;Deselect
N100 M30

Select linear and circular segmentation + parameterisation:
N30 \#SEGMENTATION ON [LIN LENGTH 0.5 CIR OPMODE 1 PARAM 0.5]
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 \#SEGMENTATION OFF ALL ;Deselect
N100 M30 ; Alternative: \#SEGMENTATION OFF [LIN CIR]

Combined selection of linear and circular segmentation:
N30 \#SEGMENTATION ON ALL
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 \#SEGMENTATION OFF ALL ; Deselect
N100 M30

### 12.32 Enlarging/reducing contours (\#SCALE ON/OFF))

The \#SCALE command can enlarge or reduce the scales of contours or positions by specifying axis-specific factors. When different factors are specified, it also permits the expansion or compression of contours.
Scaling acts on

- Path axes (linear axes, e.g. X, Y, Z, U, V, W)
- Positioning axes (independent axes, oscillating axes).

Offsets are generally not scaled, regardless of whether an offset is selected or programmed before or after \#SCALE ON.

Scaled values always refer to the zero point of the currently active coordinate system. In particular when a contour is described in absolute coordinates, you are advised, before selecting the scaling, to place the zero point on the starting point of the contour by means of the appropriate $G$ functions G53 to G57, G159, G92 or by the functions for defining the rotated coordinate systems \#ROTATION and \#CS.

```
#SCALE [ON] <axis_name><expr> {<axis_name><expr>} (modal)
#SCALE OFF (modal)
ON Activate scaling
```

OFF
<axis_name><expr> Axis-specific scaling factors. The factor must be >0.
An error message is issued if factors are $\leq 0$.

## Notice

The definition of scaling factors and their selection can be specified together or in separate steps. This means that it is possible, for example, to define the scaling factors first and then to activate scaling in a second command.
The programmed scaling factors remain stored up by M30 to program end and can be used again if \#SCALE ON/OFF is used several times.

## Attention

The scaling factors can be modified when scaling is deselected.
Modifying the scaling factors while scaling is active leads to the output of an error message.

## The following must be noted when scaling and circle programming are combined:

- Scaling circles only makes sense if the scaling factors of the axes involved in circular interpolation are identical. This applies in particular to programming circles with radius specification via $R$ or G163 because the factor for the radius is based here on the scaling factor programmed with "X".
- When circles are programmed with I, J, K, different scaling factors are also possible; - then, however, they normally cause errors in the circle centre point correction or create circle segments without any practical meaning.


## Programing Example

## Enlarge and reduce contours

## ;Scale an absolute programmed contour with identical factors

```
%L Cont1 abs
N01 G01 G90 F2000
N02 X90 Y0
N03 G301 I20
N04 X90 Y120
N05 G302 I20
N06 X50 Y120
N07 X50 Y100
N08 X40 Y100
N09 G03 X0 Y100 I-20 J0
N10 G01 X0 Y20
N11 G03 X20 Y0 R20
N13 M17
```

\%scale
N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1 abs ; Basic contour
N075 \#SCALE X0.5 Y0.5 ;Definition of scaling factors
; Definition of zero point origin of the scaled contour
N085 V.G.NP[1].V.X $=30$
NO90 V.G.NP[1].V.Y = 20
N095 G54
N100 G90 G0 X0 Y0 ZO
N105 \#SCALE ON
N110 LL Cont1 abs
N115 \#SCALE OFF
N140 M30


## ;Scale an absolute programmed contour with different ;factors

```
%L Cont1 abs
N01 G01 G90 F2000
N02 X90 Y0
N03 G301 I20
N04 X90 Y120
N05 G302 I20
N06 X50 Y120
N07 X50 Y100
N08 X40 Y100
N09 G03 X0 Y100 I-20 J0
N10 G01 X0 Y20
N11 G03 X20 Y0 R20
N13 M17
```

\%scale

N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1 abs ; Basic contour
N075 \#SCALE X0. 6 Y0. 3 ;Definition of scaling factors
; Definition of zero point origin of the scaled contour
N085 V.G.NP[1].V.X $=30$
N090 V.G.NP[1].V.Y $=20$
N095 G54
N100 G90 G0 X0 Y0 Z0
N105 \#SCALE ON
N110 LL Cont1 abs
N115 \#SCALE OFF

## N140 M30



```
;Multiple different scaling of a relative programmed
;contour
```

\%L Cont1_rel
N01 G01 G91 F2000
N02 X90 Y0
N03 G301 I20
NO4 X0 Y90
N05 G302 I20
N06 X-40 Y0
N07 X0 Y-20
N08 X-10 Y0
N09 G03 X-40 Y0 I-20 J0
N10 G01 X0 Y-50
N11 G03 X20 Y-20 R20
N13 M17
\%scale
N015 G53
NO20 G00 X0 YO Z0
; Basic contour
N030 LL Cont1 rel
N040 \#SCALE X̄̄.3 Y0.3 ;Definition 1 of scaling factors
; Definition 1 of starting point of the scaled contour
N055 G90 G0 X10 Y50
NO60 \#SCALE ON
N065 LL Cont1 rel
N070 \#SCALE OFF
N075 G90 G00 X20 Y0
N085 \#SCALE X0.5 Y0.5 ;Definition 2 of scaling factors
; Definition 1 of starting point of the scaled contour
N100 G90 G0 X40 Y10
N105 \#SCALE ON
N110 LL Cont1 rel
N115 \#SCALE OFF
N125 M30


## ;Scale a contour in a coordinate system which is shifted and rotated with \#CS

\%L Cont1_rel
N01 G01 G91 F2000
N02 X90 Y0
N03 G301 I20
NO4 X0 Y90
N05 G302 I20
N06 X-40 Y0
N07 X0 Y-20
N08 X-10 Y0
N09 G03 X-40 Y0 I-20 J0
N10 G01 X0 Y-50
N11 G03 X20 Y-20 R20
N13 M17
\%scale
N015 G53
NO20 G00 XO YO Z0
; Basic contour
N030 LL Cont1 rel
N040 \#SCALE X0. 6 Y0. 6 ;Definition of scaling factors
NO45 \#CS ON [40,5,0,0,0,20]
N055 G90 G0 X0 Y0
NO60 \#SCALE ON
N065 LL Cont1 rel
NO70 \#SCALE OFF
N0525 \#CS OFF
N125 M30

;Scale a semicircle, programmed with I, J
\%L Circle1
G02 G91 X80 Y80 I40 J40
M17
\%scale
N015 G53
NO20 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
; Basic semicircle
NO30 LL Circle1
N040 \#SCALE X0.7 Y0.7 ;Definition of scaling factors
N055 G90 G0 X10 Y10
NO60 \#SCALE ON
N065 LL Circle1
N070 \#SCALE OFF
N075 G90 G0 X0 Y0
M30

$$
\times 10^{6}
$$



## ; Scale a semicircle, programmed with $R$

\%L Circle2
G03 R33 X-66 Y0
M1 7
\%scale

N015 G53
N020 G00 X0 Y0 Z0 F1000
; Basic semicircle
N030 LL Circle2
N040 \#SCALE X1.5 Y1.5 ;Definition of scaling factors
N055 G90 G0 X10 Y10
NO60 \#SCALE ON
N065 LL Circle2
N070 \#SCALE OFF
N075 G90 G0 X0 Y0

M30

;Asymmetrical scaling of a semicircle programmed with I, J
\%L Circle1
G02 G91 X80 Y80 I40 J40
M1 7
\%scale

N015 G53
N020 G00 XO YO ZO
N025 G01 X10 Y10 F1000
; Basic semicircle
N030 LL Circle1
N040 \#SCALE X0.3 Y0. 8 ;Definition of scaling factors
N055 G90 G0 X10 Y10
NO60 \#SCALE ON
N065 LL Circle1
N070 \#SCALE OFF
N075 G90 G0 X10 Y10

M30

$$
\times 10^{6}
$$



### 12.33 Punching and nibbling

With punching and nibbling operations, automatic stroke motions are executed in a programmable number of partial steps. Users can freely define the stroke motion sequence. The stroke motion is only triggered in the active machining plane with active path motions.

### 12.33.1 Splitting up motion path and programming (\#STROKE DEF, \#PUNCH ON/ OFF, \#NIBBLE ON/OFF)

The motion path is split up either by specifying the Length of a partial segment or the number of partial segments. Programming is modal.
Splitting up by specifying the length of a partial segment takes place so that the travel block is uniformly subdivided into partial segments. The length of each real partial segment is less than or equal to the length of the programmed partial segment.
When the motion path is split up by specifying the number of partial segments, the segment length is calculated automatically.


Fig. 116: Splitting up linear blocks
Programmed circular blocks are converted to linear partial motions and splitting of the motion path refers to the arc length of the circle.


Fig. 117: Splitting up circular blocks

## Definition of stroke motion:

To define the reciprocating motion, a sequence limited to a maximum of 10 blocks is programmed, which is delimited with the following two commands:

| \#STROKE DEF BEGIN | Start of stroke definition | (modal) |
| :--- | :--- | :--- |
| \#STROKE DEF END | End of stroke definition | (modal) |

Only the following restricted scope of functions is allowed for stroke definition:
\#STROKE DEF BEGIN
G0, G01, M, H, G261, G61, G260, G60, G04, \#TIME, G90, G91
\#STROKE DEF END

Notice
The use of any other NC commands within the stroke definition leads to an error.

Activating/deactivating punching and nibbling functions:
\#PUNCH ON | OFF [ [LENGTH<expr> | NUMBER<expr>] ] (modal)
\#NIBBLE ON | OFF [ [LENGTH<expr> | NUMBER<expr>] ]
(modal)

| ON | Activate punching/nibbling |
| :--- | :--- |
| OFF | Deactivate punching/nibbling |
| LENGTH<expr> | Length of a partial segment after which a stroke motion is automatically inserted in |
|  | $[m m$, inch $]$. |
|  | With \#NIBBLE ON, a stroke motion is executed during the first path motion in the act- <br> ive machining plane, including at the start. |
| NUMBER<expr> | Number of partial segments to be generated within one motion command. A stroke <br> motion is triggered after every partial segment. |

## Notice

LENGTH and NUMBER are exclusive, i.e. splitting up is based either on partial segment length or the number of partial segments.

## Programing Example

Split up motion path and programming

```
%Nib.ble
N10 G90 G17
N2O #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50 Z-10
N60 #STROKE DEF END
N70 X10 Y30 C0
N80 #NIBBLE ON [LENGTH 5]
N90 X30 Y30 C180
N100 X30 Y10 C360
N110 #NIBBLE OFF
N120 M30
```



## Programing Example

```
%Punch
N10 G90 G17
N2O #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50 Z-10
N60 #STROKE DEF END
N70 X10 Y30
N80 #PUNCH ON [LENGTH 5]
N90 X30 Y30
N100 X30 Y10
N110 #PUNCH OFF
N120 M30
```



## Configuration:

The following setting must be entered in P-STUP-00060 in order to use this function: configuration.channel[0].path_preparation.function FCT_DEFAULT | FCT_PUNCHING configuration.channel[0].path_preparation.function FCT_DEFAULT | FCT_NIBBLING

### 12.33.2 Further functions

## Influencing the stroke triggering time:

The stroke triggering time can be influenced to achieve a good machining result. Stroke pre-triggering permits compensation of the constant dead time in signal processing. Stroke post-triggering can compensate for the transient response of the axes or, for example, for the hold-down time of pressure pads in punch operations.

Depending on the setting, influencing of the stroke triggering time refers to one of the following events:

- Interpolation target point reached $\rightarrow$ G00,G01,G02,G03
- In-Pos window of the slowest axes $\rightarrow$ G60

The dwell time G04 together with an M function is used for stroke post-triggering. An M function of the MET_SVS type is used for pre-triggering.

## Notice

The offset timing resolution depends on the CNC controller cycle time as the smallest unit.

## Example

## Stroke pre-triggering:

The user-specific M function M97 is to be output to the PLC 40 millimetres before the synchronisation time in the block sequence is reached.

Excerpt from the channel parameter list:
\# Definition of $M$ functions and synchronisation types
\# $======================================================$
m_synch[97] 0x02000000 MET_SVS
\# Setting the pre-output time, pre-output path with MET_SVS type
\# $================================================================$
m_pre_outp[97] 40000 in us

N10 \#STROKE DEF BEGIN
N30 M97
N40 \#STROKE DEF END

## Example

## </> <br> Stroke post-triggering:

M96 is the user-specific M function for triggering the stroke motion. The stroke motion should only take place 50 milliseconds after the stroke positions are reached.

N10 \#STROKE DEF BEGIN
N20 G04 0.05
N30 M96
N40 \#STROKE DEF END

## Example

## Example of output of the $\mathbf{M}$ function referred to the In-Pos window:

M96 is the user-specific M function for triggering the stroke motion. The stroke motion should only take place 50 milliseconds after the physical stroke positions of all axes are reached (actual position is in the window).

```
N10 #STROKE DEF BEGIN
N20 G04 0.05
N30 M96
N40 #STROKE DEF END
\begin{tabular}{lcc} 
G90 & G17 & \\
N10 & X10 & Y30 \\
N20 & \#PUNCH ON & [LENGTH \\
N30 & G60 X30 & Y30 \\
N40 & G60 X30 & Y10 \\
N90 & \#PUNCH OFF &
\end{tabular}
```


### 12.33.3 Restrictions

The use of rotations that tilt the active machining plane is not permitted. By contrast, rotations about the Z axis when the XY plane is active are permitted. This restriction applies only if axis motions are programmed in the stroke sequence. The output of M functions, for example, is still possible.

### 12.34 Controlling edge machining (\#EDGE MACHINING)

Depending on the machining technology it may be necessary to control the machining process especially on sharp contours (edges). In the case of a sharp edge (defined by the angle difference between two contour elements), the path velocity at the edge is modified depending on predefined parameters. Linear or circular blocks can be programmed as contour elements. No check is made here whether they are inner or outer contours.

The edge machining function is configured in the channel parameter list and is effective as of program start. The exact description of parameters is described in the documentation [MDS-CHAN// Section Settings for edge machining].

In addition, the edge machining function can also be selected/deselected and parametrised in the NC program by the following NC command:

## Selecting edge machining:

```
#EDGE MACHINING ON [ [ANGLE_LIMIT<expr>][WAIT_TIME<expr>]
                        [PRE_DIST<expr>][PRE_FEED<expr>]
    [POST_DIST<expr> ][POST_FEED<expr> ]
    [ DISABLE_FEED_ADAPTION<expr> ]] (modal)
or
#EDGE MACHINING ON [ [ANGLE_LIMIT][WAIT_TIME ]
    [PRE_DIST ][PRE_FEED ]
    [POST_DIST][POST_FEED ]
    [DISABLE_FEED_ADAPTION]] (modal)
```

or
\#EDGE MACHINING ON DEFAULT (modal)

ON
ANGLE_LIMIT<expr>

WAIT_TIME<expr>
PRE_DIST<expr>
PRE_FEED<expr>

POST_DIST<expr>
POST_FEED<expr>

DISABLE_FEED
ADAPTION<expr>

DEFAULT

Select edge machining.
Limit knee angle in [ ${ }^{\circ}$ ]. If the knee angle between two contour elements is smaller than this critical angle, the special edge function is executed.
Wait time in the edge in [s]
Distance after the edge in [mm, inch] for which a PLC signal is generated.
Feed before the edge in $[\mathrm{mm} / \mathrm{min}, \mathrm{m} / \mathrm{min}$, inch $/ \mathrm{min}$ ] used for the motion from PRE_DIST to the stop in the edge.
Distance after the edge in [mm, inch] for which a PLC signal is generated. Feed after the edge in $[\mathrm{mm} / \mathrm{min}, \mathrm{m} / \mathrm{min}$, inch $/ \mathrm{min}$ ] used for the motion up to POST_DIST. The motion then continues at the originally programmed feed rate.
Switching feed adaptation, Boolean:
0 : Feed adaptation is active
1: Feed adaptation is inactive, PRE_FEED and POST_FEED are not effective. In combination with ON selection, the default values apply from the channel parameter list [@@P-CHAN-00221-P-CHAN-00226, P-CHAN-00300].

If no values are programmed after all or some single keywords ANGLE_LIMIT, WAIT_TIME, PRE_DIST, PRE_FEED, POST_DIST, POST_FEED or DISABLE_FEED_ADAPTION, the default values from the channel parameter list [P-CHĀN-00221-P-CHAN-00226, P-CHAN-00300] are used.

The command \#EDGE MACHINING OFF deselects the edge machining function. In combination with deselection, no other parameters may be specified.

## Deselecting edge machining:

```
#EDGE MACHINING OFF
(modal)
```


## Configuration:

To use this function, the following setting must be made in the start-up list ([STUP]): configuration.channel[0].path_preparation.function FCT_DEFAULT | FCT_EMF

## Programing Example

Control edge machining

```
%edge_machining
(Select edge machining and set parameters with value)
#EDGE MACHINING ON [ANGLE_LIMIT=90 WAIT_TIME=0.2 PRE_DIST=5 PRE_FEED=800
POST_DIST=10 POST_FEED=1600]
(Select edge machining and set parameters to default)
#EDGE MACHINING ON [ANGLE_LIMIT PRE_DIST PRE_FEED POST_DIST POST_FEED
WAIT_TIME DISABLE_FEED_ADAPTION]
or
(Select edge machining and set parameters to default)
#EDGE MACHINING ON DEFAULT
(Deselect edge machining)
#EDGE MACHINING OFF
```


### 12.35 Switching dynamic weighting (\#DYNAMIC WEIGHT)

Path and radius dependent weighting of dynamic limits are configured in the channel parameter list and activated by parameters P-CHAN-00190 and P-CHAN-00230. Configuration examples for path-dependent weighting and radius-dependent weighting are contained in [CHAN].
The weightings are then active after control start-up.
The following NC command can switch the dynamic weightings during NC program runtime, i.e. they can be activated and deactivated as required.

```
#DYNAMIC WEIGHT [ON | OFF] [ [PATH | CURVE] ] (modal)
```

| ON | Weighting on |
| :--- | :--- |
| OFF | Weighting off |
| PATH | Path length dependent weighting (see P-CHAN-00190) |
| CURVE | Circle radius dependent weighting (see P-CHAN-00230) |

## Notice

For activation or deactivation, at least one keyword PATH or CURVE must be programmed.

## Programing Example

## Switch dynamic weighting

```
N20 G00 X0 Y0 Z0 F10000
N3O #DYNAMIC WEIGHT ON [PATH] ;Select path-dependent dynamic weighting
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 #DYNAMIC WEIGHT OFF [PATH] ;Deactivate
N100 M30
```


### 12.36 Weighting of external feedrate (\# FF)

The NC channel includes the option of commanding an external path feed via the PLC [HLI// Control commands of a channel]. This can be additionally weighted in the NC program by the feed factor \#FF.

The effect of \#FF is controlled by setting the parameter P-CHAN-00422.

```
#FF = <feed_factor> (modal)
```

<feed_factor>

Weighting of external feed in [\%]

## Notice

The feed factor \#FF only acts on an external specified feedrate.
Before Build V3.01.3079.0, the feed factor \#FF is limited to maximum $10000 \%$ and minimum to $>0$. As of Build V3.01.3078.0, it is limited to maximum $1000000 \%$ and minimum to $0.1 \%$.

The NC command is available as of CNC Build V3.01.3064.02.

### 12.37 Axis clamping and monitoring

To ensure high quality and accuracy in a machining process (milling, turning, erosion machining, etc.), the tool-carrying axis can by physically fixed by the drive brake. Mechanical axis clamping is commanded by synchronised M/H functions in the NC program and enabled in the PLC. This ensures that an axis is clamped before the next motion block is executed.

The status 'clamped axis' is effective at axis driver level and remains valid cross-channel until it is deselected, even with axis exchange operations.
While an axis is clamped, neither the interpolator nor an additional interface can command it to drive by command values. The monitor required here is enabled and disabled by the following NC command. If a clamped axis is to be moved, the error message P-ERR-70525:is output and the CNC changes to error state.

The monitoring status is modal and remains active even after NC program end or RESET.
Monitoring specific clamped axes:
\#CLAMP MONITORING ON | OFF [ \{ AX=.. | AXNR=.. \} ]
AX=<axis_name> Name of clamped axis to be monitored.
AXNR=<expr> Logical number of the axis to be monitored, positive integer

Monitoring all clamped axes:
\#CLAMP MONITORING ON | OFF ALL

## Programing Example

Axis clamping and monitoring

```
%clamp.nc
N010 X0 Y0 Z0 A0 BO C0
```

```
N020 A[M300] C[M300] ;Mechanical clamping of axes A + C on
NO30 #CLAMP MONITORING ON ALL ; Enable monitoring of axes A + C
N100 #CLAMP MONITORING OFF ALL ; Disable monitoring of axes A + C
N110 #CLAMP MONITORING ON [AX=A AX=C] ; Enable monitoring of axes A + C
N120 Y10
N130 X15
N140 #CLAMP MONITORING OFF [AX=A AX=C] ;Disable monitoring of axes A + C
N150 X10
N160 #CLAMP MONITORING ON [AXNR=4 AXNR=6] ; Enable monitoring of axes A + C
N120 Y20
N130 X25
N150 A[M301] C[M301] ; Release mechanical clamping of axes A + C
N140 #CLAMP MONITORING OFF [AXNR=4 AXNR=6] ; Disable monitoring of axes A + C
N999 M30
```


## Notice

i
The command does not cause any mechanical axis clamping. Instead, it enables the monitor for unauthorised axis motions. With the function 'Backward motion on path', the monitor status is inverted accordingly.

## Notice

Axis motions which are generated by active compensation functions in the axis driver (e.g. Cross compensation, Volumetric Compensation, etc.) are not monitored.

The monitoring status of an axis can be read by the ADS access..
Index group: 0x20300
Index offset: 0x10189 (axis 1)

## $12.38 \quad$ Gantry start-up

## Attention

Only use the \#GANTRY command for start-up
Possible machine damage if command used incorrectly.

## Disable gantry combination

```
#GANTRY OFF [ { AXNR=.. | AX=.. }]
```

| AXNR=<expr $>$ | Logical axis number (P-AXIS-00016) of master axis |
| :--- | :--- |
| AX=<Namemasteraxis $>$ | Name of the master axis of a gantry combination |

## Disable all gantry combinations:

```
#GANTRY OFF ALL
```


## Restore a gantry combination

```
#GANTRY ON [ { AXNR=.. | AX=.. }]
```

AXNR=<expr>
AX=<Namemasteraxis>

Logical axis number (P-AXIS-00016) of master axis
Name of the master axis of a gantry combination

## Restore all gantry combinations:

## \#GANTRY ON ALL

For more information, see [FCT-C11// Gantry start-up]

### 12.39 Position controller-based axis couplings (\#GEAR LINK)

The movement of a target axis by a position controller-based axis coupling is influenced additively or even exclusively by the motions of other axes. The target axis is either an additional interpolator axis or a spindle.
Axes that influence a target axis via coupling specifications are referred to as source axes. The motion part of sources axes are weighted accordingly by defining specific factors. This permits the implementation of an electronic gear.


Fig. 118: View of the difference between source and target axis

This function can be controlled by the HLI and is described in detail in [FCT-A9]. The NC command \#GEAR LINK parameterises and enables this type of axis coupling, even in a subroutine.

GEAR-LINK programming syntax
Parameterisation by axis name:
\#GEAR LINK [ TARGET=.. AX<i>=.. NUM<i>=.. DENOM<i>=.. \{ AX<i>=.. NUM<i>=.. DENOM<i>=.. \}
[ MODE=.. ] [ ACC=..] \{ <br>$] }$
(modal)

```
TARGET=<axis_name> Name of coupling target axis
AX<i>=<axis_name> Name of source axis i where i =1 .. 4
NUM<i>=<expr>
DENOM<i>=<expr>
MODE=<mode>
ACC=<expr> Acceleration limit on the axis with coupling/decoupling in [mm/\mp@subsup{sec}{}{2}, inch/\mp@subsup{sec}{}{2}]. If ACC
    is not specified, P-AXIS-00053 is used as default acceleration limit.
    Separator ("backslash") for clear programming of the command over multiple lines.
```

Parameterisation by axis number:

```
#GEAR LINK [ TARGETNR=.. AXNR<i>=.. NUM<i>=.. DENOM<i>=.. { AXNR<i>=.. NUM<i>=..
    DENOM<i>=..} [ MODE=..] [ ACC=..] {\}]
    (modal)
```

TARGETNR=<axis_number>
AXNR<i>=<axis_number>
NUM<i>=<expr>
DENOM<i>=<expr>
MODE=<mode>

ACC=<expr>
1

Logical number of coupling target axes, positive integer
Logical number of source axis i where i=1 .. 4, positive integer
Coupling factor NUM[i] / DENOM[i] for source axis i;
numerator and denominator are negative or positive integers.
Mode for coupling:
DIRECT: Coupling requires all the axes involved to be at standstill (default). If this is not the case a stop occurs with error message P-ERR-70200
SOFT: Soft coupling using slope; axes can move. If ACC is not specified, P-AXIS-00053 is used as default acceleration limit.
Acceleration limit on the axis with coupling/decoupling in [mm/sec ${ }^{2}$, inch/sec ${ }^{2}$ ] Separator ("backslash") for clear programming of the command over multiple lines.

Enabling and parameterising in one step:
\#GEAR LINK ON [ .. ]
...or disable only
A previously parameterised coupling is enabled. An error message is output if no coupling was parameterised.
\#GEAR LINK ON [ TARGET=.. | TARGETNR=.. ]
...or disable only:
\#GEAR LINK OFF [ TARGET=.. | TARGETNR=.. ]

## Response at program end and reset:

Active axis couplings are disabled at program end and reset.

## Programing Example

## Defining and selecting a coupling specification

```
Parameterisation by axis name:
#GEAR LINK [TARGET=U AX1=X AX2=Y NUM1=1 DENOM1=2 NUM2=-1 DENOM2=1\
    MODE=SOFT ACC=400]
:
TARGET=U ;Target axis is U AXIS
AX1=X ; Source axis 1 is X axis
NUM1=1 DENOM1=2 ;Coupling factor for X axis is NUM1/DENOM1=0.5
AX2=Y ;Source axis 2 is Y axis
NUM2=-1 DENOM2=1 ; Coupling factor Y axis is NUM2/DENOM2=-1
MODE=SOFT ; Soft coupling/decoupling
ACC=400 ;Acceleration limit at 400 mm/\mp@subsup{sec}{}{2}
Enable coupling:
#GEAR LINK ON [TARGET=U]
Disable coupling:
#GEAR LINK OFF [TARGET=U]
Alternative parameterisation by axis number:
#GEAR LINK [TARGETNR=4 AXNR1=1 AXNR2=2 NUM1=1 DENOM1=1 NUM2=-1 \
    DENOM2=1]
Enable coupling:
#GEAR LINK ON [TARGETNR=4]
Disable coupling:
#GEAR LINK OFF [TARGETNR=4]
```


### 12.40 Settings for turning functions (\# TURN)

Release Note
This function is available as of CNC Build V3.1.3079.03.

The \#TURN command can influence rotary functions.
Syntax:
\#TURN [ [ROT_FEED_CPL=..]] modal

ROT_FEED_CPL=.. Influence of axis couplings on the revolution feedrate with G95 [ 637]
0 : Axis couplings are not considered (default)
1 : Axis couplings are considered

### 12.41 Distance to go display in a program section



## Release Note

This function is available as of CNC Build V3.1.3079.27.

A geometry sequence can be flagged in the NC program with the NC commands \#DIST TO GO BEGIN/END. Within a sequence, the distance to go of each axis is displayed on the HLI up to the end of the geometry sequence by dist_to_geom_end.
Requirement: The parameter P-STUP-00033 must be parameterised in order to use the display function.

## \#DIST TO GO BEGIN <br> \#DIST TO GO END

Characteristics:

- The distance to go of the axis is always displayed in forward direction. Backward motion on the path makes the distance displayed greater.
- The displayed values are no longer correct if a geometry changes takes place with the "Delete distance to go" function. This error is signalled by the warning ID 51047.
- The entire geometry sequence is required to obtain a valid display of the axis distances to go up to \#DIST TO GO END.
Especially with large program sequences and small values defined in P-STUP-00033, this may result in a delay of the display datum dist_to_geom_end and the warning ID 51048.
If the values in P-STUP-00033 are too small, the sequence is output without stopping and the display datum dist_to_geom_end remains at 0 until a valid value can be determined for the display.
- If NC commands that result in a channel synchronisation are programmed within the flagged geometry sequence. The commands prevent the look-ahead distance to go calculation up to \#DIST TO GO END. The distance to go display is then only displayed correctly after this NC command.
Example of these NC commands:
- \#CHANNEL INIT [ ${ }^{\text {177 }}$ ]
- \#FLUSH WAIT [> 339]
- or reading a synchronous V.E. variable


## Programing Example

## Distance to go display in a program section

```
N010 G01 X0 Y0 F5000
N020 #DIST TO GO BEGIN
N030 G01 X10
N040 G01 Y10
N050 G01 X5
N060 #DIST TO GO END
```

In the example, the display would show at the start of the sequence for X 15 and Y 10 and 0 in each case at the end.

The display values is presented in the figure below:


Fig. 119: Distance to go display in a program section

## 13 Tool geometry compensation (D)

Tool geometries are compensated for length, radius and axis offset. The corresponding compensation data for tool geometry are provided either by:
tool data lists containing data records for each tool and loaded at controller start-up (see [TOOL]), or an external tool management system.

## Tool compensation data

Tool compensation data is selected using the D word or is selected automatically by the T word (P-CHAN-00014) if this is parameterised.
D <expr> $\quad$ Select tool compensation $\quad$ (modal)
<expr> Number of the tool compensation data block. Positive integer.
The D word defines a tool data block that contains the following values:

- Tool length (perpendicular to main plane)
- Tool radius (in the main plane)
- Axis offset coordinates (in all axis directions)
- Measuring unit of the numerical statements (mm/inch)
- Tool validity code
- Tool dynamic data


## Attention

The time when the tool compensation data for the tool length (perpendicular to the main plane) and the axis offset coordinate (in the axis directions) become effective is programmed by the channel parameter P-CHAN-00100. The lines below must also be noted:

A distinction is made between two modes:
The compensating movement is executed directly with the D word without specifying a path preparatory function.
For safety reasons, the controller only executes a compensating motion at the next absolute motion block in the corresponding axis (default mode).
In G91 mode, the following applies to the inclusion of tool compensation data:
The programming of...
N10 D16
N20 G0 X0 G91
... may not cause any movement of the X axis (corresponds to relative movement about 0 ). In this case, tool compensation data for this axis only acts if the next motion information is programmed in absolute coordinates (G90).
Please note for both modes together: The tool radius is transferred to the tool radius compensation and influences the equidistant calculation. The compensating movement always take place combined with a path preparatory function.
The rules for execution of the compensating movement also apply to deselection of tool compensation.

```
D00
    Deselecting tool compensation
```


### 13.1 Tool length compensation

On Cartesian machines, tool length compensation (TLC) always acts in the direction of the 3rd main axis. With G17, this is generally the tool head axis.

If the tool is oriented in the negative direction to the $Z$ axis, a compensating motion takes place to the positive direction of the $Z$ axis.


Fig. 120: Compensate tool length by compensating motion

## Attention

If the mechanical machine structure defines a different tool head axis, the user must choose (G18, G19, \#CS, \#TRAFO) a suitable programming coordinate system (PCS) in order to calculate the tool length correctly.

In the example below, tool length compensation is carried out in the $Z$ direction. When the compensation data block D16 is selected in block N30, the compensation movement occurs in the $Z$ direction jointly with the motion data block in N30.

## Programing Example

```
N10 G01 F900 G17 ; X-Y plane, length compensation in Z+ (default))
N20 X150 Y10 Z10
N30 D16 Y40 Z15 ; Select length compensation D16. The
: ;compensation motion is executed.
:
N100 D20 ; Select tool compensation D20. The
: ;compensation motion only occurs with
: ; the next absolute motion block in the Z direction.
N200 G0 D0 X0 Y0 ZO ; Deselect TLC
Compensation data block
D0: Length = 0 Radius = 0
D16: Length = 5 Radius = 5
D20: Length = 12.5 Radius = 5
```

Compensatory movement in the Z-direction for N30


Fig. 121: Example of tool length compensation

### 13.1.1 Axis-specific assignment of tool length compensation (\#TLAX, \#TLAX DEFAULT)

In initial state, the tool length is always considered in the third main axis of the new place when the plane changes (G17, G18, G19). This is useful if the tool head (or tool axis) is orientable and machining takes place in the relevant plane.

This behaviour is not required with non-orientable (fixed) tool heads. If a change in place is only intended to program an approach motion, the tool length and its orientation can remain in the original main axis or can be assigned fixed to a specific main axis by the following NC command:

```
#TLAX [ <axis_name> + | - ] (modal)
```

<axis_name> Name of main axis in which tool length is included and orientation is specified

The following rule applies to orientation assignment:


Fig. 122: Assignment rule of tool length compensation

In initial state (G17), tool length compensation is included in the third (Z). The direction (orientation) is specified by the sign ' + '. This corresponds in the $\mathrm{X}-\mathrm{Y}$ plane to the command \#TLAX [Z+].

On machines with non-orientable tool heads that are mainly used for 3D machining in G17, this behaviour can also be preset by the channel parameter P-CHAN-00420; it is then not necessary to program \#TLAX.

The command below cancels the fixed assignment of tool length compensation to a specific axis. Tool length is then considered in the third main axis of the current plane again.

## Programing Example

```
Contour circular motions in different planes with tool length compensation in con-
stant orientation
%tlax
NO10 G0 X0 Y0 Z0
N020 V.G.WZL=33 G161
N030 X0 Y0 Z0
NO40 #TLAX [Z+] ;Tool length compensation in Z+
N050 G18
N060 G01 X0 Z50 F2000
N070 G02 X50 Z0 I50 K50 F1000
N080 G17
N090 G03 X100 Y50 I50 J50
N100 G19
N110 G03 Y0 Z50 J50 K50 F1000
N120 G18
N130 G02 X0 Z50 I50 K50 F1000
N140 #TLAX DEFAULT ; Deselect tool length compensation in Z+,
    ;Tool length is included in Y (G18)
N150 G17
N160 M30
```



Fig. 123: Contour approach motions in G17, G18, G19 with tool length compensation in constant orientation

### 13.2 Tool radius compensation (TRC)

Tool radius compensation (TRC) allows programming of the workpiece contour independently of tool geometry. If TRC is selected (G41, G42), a tool path equidistant from this programmed tool contour is calculated at distance "tool radius".

Tool radius compensation acts in the plane selected with G17, G18 or G19. The tool compensation values used are the tool compensation synchronisation modes stored under the D words (see Tool geometry compensation [> 492]).
Tool radius can also be changed when TRC is active by selecting a new $D$ word or writing the variable V.G.WZ_AKT.R.
When a negative tool radius is used, the selection side of the TRC is changed automatically.


Fig. 124: Mode of operation and terms of tool radius compensation

## Overview of all TRC-relevant G functions:

## Selecting/deselecting TRC

| G40 | Deselecting TRC | (modal, initial state) |
| :--- | :--- | :--- |
| G41 | Selecting TRC left of contour | (modal) |
| G42 | Selecting TRC right of contour | (modal) |

## Selecting/deselecting TRC

| G138 [ 507] | Direct TRC selection/deselection | (modal) |
| :---: | :---: | :---: |
| G139 [> 513] | Indirect TRC selection/deselection | (modal, initial state) |
| G236 [ ${ }^{\text {540] }}$ | Direct TRC selection/deselection on the path | (modal) |
| G237 [>525] | Perpendicular TRC selection/deselection | (modal) |
| G238 [> 532] | Selecting inside corner of TRC | (modal) |
| G239 [ 535] | Selecting/deselecting TRC directly without block | (modal) |
| G05 [ 556] | Tangential TRC selection/deselection | (non-modal) |

## Attention

It is not permitted to change the modal selection method to $G 238$ when TRC is selected.

## Feed adaption:

| G10 [ 5559$]$ | Constant feedrate | (modal, initial state) |
| :--- | :--- | :--- |
| G11 [ $>559]$ | Adapted feedrate | (modal) |

## Contour masking:

| G140 [> 561] | Deselecting contour masking | (modal, initial state) |
| :--- | :--- | :--- |
| G141 [ 561] | Selecting contour masking | (modal) |

## Selecting external transitions:

Outside corners must be bypassed if tool radius compensation is selected. For this purpose, tool radius compensation inserts transition blocks. Use G25 and G26 to select between linear block insertion and circular block insertion.

| G25 | Linear transition | (modal, initial state) |
| :--- | :--- | :--- |
| G26 | Circular transition | (modal) |

## Programing Example

Presentation of the different selection options when tool radius compensation is used. This shows the commands for direct, indirect and tangential selection/deselection of TRC (G139/G138/G05) in combination with the TRC transition types for linear and circular transition (G25/G26).
The test programs are executed at a tool radius of 10 mm .

| Example 1 | G138 | G139 | G05 |
| :---: | :---: | :---: | :---: |
| G25 | Path 1 | Path 2 | Path 3 |

```
%WZKG25 (Contour for G25)
N1 G00 G90 T1 D1 X0 Y0 Z0 G17
(Display of contour)
N15 G01 X20 Y20 F1000
N20 G91
N25 G1 X10
N30 X5 Y-5
N35 Y-5
N40 X-5 Y-3
N45 G01 G90 X0 Y0 F1000
```

(Path 1)

| N100 G138 G41 | (Select directly and TRC left of contour) |
| :--- | :--- |
| N105 G01 X20 Y20 F1000 | (Required compensation motion after G41) <br> N110 G25 |
| N115 G1 G91 X10 |  |
| N125 linear transitions) |  |
| N125 Y-5 |  |
| N130 X-5 Y-3 |  |
| N135 G138 G40 |  |
| N140 G01 G90 X0 Y0 F1000 |  |

(Path 2)

```
N200 G139 G41 (Select directly and TRC left of contour)
N205 G01 X20 Y20 F1000 (Required compensation motion after G41)
N210 G25 (G25 linear transitions)
N215 G1 G91 X10
N220 X5 Y-5
N225 Y-5
N230 X-5 Y-3
N235 G139 G40 (Deselect indirectly and deselect TRC)
N240 G01 G90 X0 Y0 F1000 (Required compensation motion after G40)
```

(Path 3)

```
N300 G05 G41 (Tangential selection and TRC left of contour)
N305 G01 X20 Y20 F1000 (Required compensation motion after G41)
N310 G25
N315 G1 G91X10
N320X5 Y-5
N325 Y-5
N330 X-5 Y-3
N335 G05 G40 (Tangential deselection and deselect TRC)
N340 G01 X20 Y20 F1000 (Required compensation motion after G41)
N999 M30
```

(Tangential selection and TRC left of contour)
(Required compensation motion after G41)
(G25 linear transitions)


| Example 2 | G138 | G139 | G05 |
| :--- | :--- | :--- | :--- |
| G26 | Path 4 | Path 5 | Path 6 |

```
%WZKG26 (Contour for G26)
N10 G00 G90 T1 D1 X0 Y0 Z0 G17
(Display of contour)
N15 G01 X20 Y20 F1000
N20 G91
N25 G1 X10
N30 X5 Y-5
N35 Y-5
N40 X-5 Y-3
N45 G01 G90 X0 Y0 F1000
```

(Path 4)

| N400 G138 G41 | (Select directly and TRC left of contour) <br> N405 G01 X20 Y20 F1000 <br> N410 G26 |
| :--- | :--- |
| (Required compensation motion after G41) <br> N415 G1 G91 X10 <br> N420 X5 Y-5 |  |
| (G26 Circular transitions) |  |
| N430 Y-5 X-5 Y-3 |  |
| N435 G138 G40 |  |
| N440 G01 G90 X0 Y0 F1000 |  |

(Path 5)

```
N500 G139 G41 (Select directly and TRC left of contour)
N505 G01 X20 Y20 F1000 (Required compensation motion after G41)
N510 G26 (G26 Circular transition)s
N515 G1 G91 X10
N520 X5 Y-5
N525 Y-5
N530 X-5 Y-3
N535 G139 G40 (Deselect indirectly and deselect TRC)
N540 G01 G90 X0 Y0 F1000 (Required compensation motion after G40)
```

(Path 6)

```
N600 G05 G41 (Tangential selection and TRC left of contour)
N605 G01 X20 Y20 F1000 (Required compensation motion after G41)
N610 G26
N615 G1 G91X10
N620 X5 Y-5
N625 Y-5
N630 X-5 Y-3
N635 G05 G40 (Tangential deselection and deselect TRC)
N640 G01 X20 Y20 F1000 (Required compensation motion after G41)
```

N999 M30


## Change tool data



## :

N30 G0 D0 X0 Y0 Z0 G17 (X-Y plane)
N40 G0 D100 X10 Y10 (Select TLC)
N50 G1 Z0
N60 G0 Z100
N70 G41 (Select TRC with data block D100)
N80 G1 Z0
N90 G2 X10 Y10 I-15 (Full circle with radius 15)
N100 G0 Z100
N110 D2 Z200 (Other compensation data block, i.e.)
(other values for TLC and TRC)
N120 G1 Z0 (Compensating motion)
(motion of TLC takes place here)
N130 G1 X20 Y20
(Compensating motion of TRC)
N140 G02 X10 Y10 I-15 (TRC is now executed with data)
(block D2)
N150 G0 Z100
N160 G40 (Deselect TRC)
N170 X0
:

## Dynamic change of tool radius:

Another option to change tool radius is to assign variables (see also Chap. 13 [> 564]). For example, this takes into consideration the wear and tear of grinding tools during motion blocks.

## Programing Example

Dynamic change of tool radius:

```
N00 G1 G90 X0 D6 F5
N10 G41 G26 (Select TRC)
N20 X0 Y250 (Starting point)
N30 V.P.VERSCHLEISS = 0.010
N100 $FOR V.P.LAUF = 0,100,10 (Grinding cycle)
N110 X300
N120 Y200
N130 X0
N140 Y250 (Tool radius gradually becomes smaller)
N150 V.G.WZ AKT.R = V.G.WZ AKT.R - V.P.VERSCHLEISS
N160 $ENDFOR
N200 G40 X300
(Deselect TRC)
N999 M30
```


### 13.2.1 Direct/indirect deselection of TRC

When TRC is selected directly or indirectly, an approach block is generated in the selected compensation plane with the next motion block in the NC program.
When TRC is selected, G138 selects an alternative approach strategy (direct selection of TRC) besides the standard approach strategy (indirect selection of TRC).

The figures below show all the possible approach blocks for the permitted contour transitions. Two NC blocks NC10 and NC20 are illustrated for the 3 relevant contour transition angles.

### 13.2.1.1 Direct selection (G138/G41/G42)

Combination type 1

Contour transition range $0^{\circ}<\beta \leq 180^{\circ}$
Combination type 1

Contour transition range $180^{\circ}<\beta \leq 270^{\circ}$
Combination type 1

Contour transition range $270^{\circ}<\beta \leq 360^{\circ}$

### 13.2.1.2 Indirect selection (G139/G41/G42) with G25

Combination type 1

Contour transition range $0^{\circ}<\beta \leq 180^{\circ}$
Combination type 1

Contour transition range $180^{\circ}<\beta \leq 270^{\circ}$
Combination type 1

Contour transition range $270^{\circ}<\beta \leq 360^{\circ}$

### 13.2.1.3 Indirect selection (G139/G41/G42) with G26

Combination type 1

Contour transition range $0^{\circ}<\beta \leq 180^{\circ}$
Combination type 1

Contour transition range $180^{\circ}<\beta \leq 270^{\circ}$
Combination type 1

Contour transition range $270^{\circ}<\beta \leq 360^{\circ}$

### 13.2.2 Direct/indirect deselection of TRC

When TRC is selected directly or indirectly, an exit block is generated in the selected compensation plane with the next motion block after a programmed G40.
When TRC is deselected, G138 can select an alternative exit strategy (direct deselection of TRC) besides the standard exit strategy (indirect deselection of TRC).

The figures below show all the possible exit blocks with all contour transitions. Two NC blocks NC10 and NC20 are shown for the 3 relevant contour transition angles.

### 13.2.2.1 Direct deselection (G138/G40)

Combination type 1

Contour transition range $0^{\circ}<\beta \leq 180^{\circ}$
Combination type 1

Contour transition range $180^{\circ}<\beta \leq 270^{\circ}$
Combination type 1

Contour transition range $270^{\circ}<\beta \leq 360^{\circ}$

### 13.2.2.2 Indirect deselection (G139/G40) with G25

Combination type 1

Contour transition range $0^{\circ}<\beta \leq 180^{\circ}$
Combination type 1

Contour transition range $180^{\circ}<\beta \leq 270^{\circ}$
Combination type 1

Contour transition range $270^{\circ}<\beta \leq 360^{\circ}$

### 13.2.2.3 Indirect deselection (G139/G40) with G26

Combination type 1

Contour transition range $0^{\circ}<\beta \leq 180^{\circ}$
Combination type 1

Contour transition range $180^{\circ}<\beta \leq 270^{\circ}$
Combination type 1

Contour transition range $270^{\circ}<\beta \leq 360^{\circ}$

### 13.2.3 Perpendicular selection/deselection of TRC (G237)

If the perpendicular mode of selection and deselection is used, there are no restrictions concerning the sequence of blocks as for the direct or indirect selection and deselection mode.
It is also possible to activate TRC for single blocks.
When perpendicular TRC is selected, a block is added running orthogonally to the programmed path. This block is output before the first motion block and represents the distance (tool radius) of the compensated path relative to the programmed path.

When TRC with the perpendicular mode is deselected, a block is added after the deselection block and runs orthogonally to the programmed path. This motion block reverses the distance generated to the programmed path.

## Programing Example

## Perpendicular selection and deselection (G237) of TRC

```
%bsp01.nc
N10 G00 X0 Y0 Z0 G17
N20 X10Y10
N30 V.G.WZ_AKT.R=5 (Tool radius)
N40 G237
(perpendicular selection activated)
; Corrected path
N50 G42 (Select TRC right)
N60 G03 X60Y60J50F1000
N70 G01 Y100
N80 G03 X10Y150J50
N90 G40 (Deselect TRC)
; Presentation of the original contour
N100 G00 X0Y0
N110 X10Y10
N120 G03 X60Y60J50F1000
N130 G01 Y100
N140 G03 X10Y150J50
N150 GOO XOYO
N999 M30
```



Fig. 125: Contour example with G237

### 13.2.3.1 Technology functions

The placement of technology functions in the NC program is especially important with perpendicular selection or deselection since this determines the output time.

When perpendicular TRC is selected, the arrangement of selection block, technology function and the first motion block is decisive for the output time.

When perpendicular TRC is deselected the arrangement of technology block and the last compensated motion block is decisive for the output time.

The programming examples below and corresponding figures illustrate this.

Programing Example

```
%bsp02.nc
N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5 (Tool radius)
N40 G237 (perpendicular selection activated)
N50 G91 (relative programming)
N60 G01 X30 Y10
; Corrected path
N50 G41 M7 X20 Y70 (Select TRC left)
N55 M8
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N85 M9
N90 G40 (Deselect TRC)
N100 G90 X200 Y0 (absolute programming)
N110 X0
; Presentation of the original contour
N200 G91 (relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0 (absolute programming)
N270 G00 X0
N999 M30
```

The technology function M9 in the figure below is output immediately before before the output of the perpendicular deselection block. The technology function must be placed between the last block to be compensated and the perpendicular deselection of TRC.


Fig. 126: Contour example with technology function 1

## Programing Example

## Technology function 2

\%bsp03.nc

```
N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5 (Tool radius)
N40 G237
    (perpendicular selection activated)
N50 G91
N60 G01 X30 Y10
; Corrected path
N50 G41 M7 (Select TRC left)
N55 M8 X20 Y70
N60 M9 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40 (Deselect TRC)
N100 G90 X200 Y0 (absolute programming)
N110 X0
; Presentation of the original contour
N200 G91 (relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0 (absolute programming)
N270 G00 X0
N999 M30
```

The technology function M8 is executed after the selection block generated. For this placement, it is absolutely necessary that it is not placed in the same block as TRC selection and that there is no motion block between the selection block and the technology function.


Fig. 127: Contour example with technology function 2

## Programing Example

## Technology function 3

```
%bsp04.nc
N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5 (Tool radius)
N40 G237
N50 G91
N60 G01 X30 Y10
; Corrected path
N50 G41 (Select TRC left)
N51 M7
N52 M8
N53 M8
N55 X20 Y70
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40 (Deselect TRC)
N100 G90 X200 Y0 (absolute programming)
N110 X0
; Presentation of the original contour
N200 G91 (relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0 (absolute programming)
N270 G00 X0
N999 M30
```



Fig. 128: Contour example with technology function 3

### 13.2.3.2 Technology function in single block

## Programing Example

Technology function in single block
\%.bsp05.nc
N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5 (Tool radius)
N40 G237
N60 G01 X10 Y10
; Corrected path
N50 G41 M7 (Select TRC left)
N55 M8 X70 Y30
N60 M7
N90 G40 (Deselect TRC)
N100 G90 X100 Y0
N110 X0
; Presentation of the original contour
N210 G01 X10 Y10
N220 X70 Y30
N240 X100 Y0
N270 G00 X0
N999 M30


Fig. 129: Contour example with technology function in single block

### 13.2.4 Selecting inside corner of TRC (G238)

Inside corner selection means that the TRC should be selected at an inside corner of a closed contour.

The designations below are used in connection with inside corner selection:

- Approach block:: linear motion block which starts at the point of selection
- 1st motion block: Motion block that starts at the end of the approach block. It is also the 2nd motion block after selection of TRC with motion data
- last motion block: last motion block before deselection of TRC with motion data


## Programing Example

## Inside corner selection (G238) of TRC

In star-shaped contours the TRC should be selected in a pointed inside corner.

```
%musterstern.nc
N1 G74 X1Y2Z3
N2 G17 G00 X0Y0Z0 G90
N4 F10000
(Display of contour)
N100 G01 X0 Y100
N110 X-20 Y20
N120 X-100 Y0
N130 X-20 Y-20
N140 X0 Y-100
N150 X20 Y-20
N160 X100 Y0
N170 X20 Y20
N180 G01 X0 Y100
N2O0 GOO XOY0
(Path display)
N210 G238 (Inside corner selection activated)
N220 V.G.WZR = 10 (Define tool radius)
N230 G41 (Select TRC left of contour)
N240 G01 X0 Y100 (Approach block)
N310 X-20 Y20
N320 X-100 Y0
N330 X-20 Y-20
N340 X0 Y-100
N350 X20 Y-20
N360 X100 Y0
N370 X20 Y20
N380 G01 X0 Y100
N390 G40 (Deselect TRC)
N400 G00 XOYO
N999 M30
```



Fig. 130: Contour example for inside corner selection (G238)

## Note:

The selection command G41 takes place at point P1; point P2 is the end point of the first linear block after G42 and must simultaneously be the target point of the last motion block before G40.

This outer (or blue) line marks the programmed contour; the inner (or red) line shows the tool path.

The closed contour means that the end point of the last motion block before TRC is deselected must be identical with the end point of the linear block after selection.

There are no restrictions for block transition between the last and the first contour motion block, neither for the transition angle nor with block combinations.

## Notice

After the programmed G40 to deselect TRC, contour violations are not monitored in the subsequent motion block.

### 13.2.4.1 Restrictions of inside corner selection

- The programmed contour must be closed.
- The last motion block may not be a full circle.
- The first motion block must be a linear block; this also applies to the first motion block after TRC is deselected.
- The point P1 must at least have the distance of the tool radius to all contour elements.
- The first motion block may not cross a contour element, with the exception of the first and the last motion blocks.
- It is not permitted to change the selection side in selected state.
- It is not permitted to make changes to the tool radius of tool in selected state.
- The selection side of the TRC must be selected from the position of selection soy that the tool does not cross the programmed contour.
- The enclosed angle of the first and the second motion blocks after selection may not exceed $180^{\circ}$.

Causes of angle restriction:


In the above graphic, the TRC should be selected to the right of the contour. As of a transition angle of over $180^{\circ}$ between the last and the first contour blocks, the first point of the equidistant path is determined using the direct selection method. This is P3. The diagram illustrates that the direct connection P1 to P3 violates the contour marked in red.

The angle shown as the starting angle in the graphic may be maximum $180^{\circ}$ in order not to damage the contour. The alternative point of selection $\mathrm{P} 1^{*}$ prevents contour damage.

## Attention

If one of the restrictions is violated, an error is output.

### 13.2.5 Direct selection/deselection of TRC without block (G239)

The approach motion with G239 is executed immediately as with G138.
When TRC is deselected (G40) the last equidistant point is the current position. A subsequent motion block to reduce the tool radius is not required.

## Attention

An arc programmed directly after G40 causes an error message because then the real deselection position and the programmed starting point of the circle are no longer identical.

In the examples below the programmed path is displayed in black and the equidistant path is displayed in blue.

## Programing Example

## Example 1

Then after TRC is deselected, a position is programmed in both main axes and executed in this way.

```
%G239 Demo1.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ AKT.R = 10
N040 G41 G01 X30 Y20 (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)
N100 G00 X150 Y0
N110 M30
```



Fig. 131: Motion block sequence of G239_a position in both main axes

## Programing Example

## Example 2:

Here only one of the main axes is programmed after TRC is deselected. The position of the 2nd main axis remains the same.

```
%G239 Demo2.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
NO30 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20 (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
NO80 G40 (Deselect TRC without block)
N090 G00 X150
N100 G00 X200
N110 M30
```



Fig. 132: Motion block sequence of G239_only one main axis programmed

## Programing Example

## Example 3:

When TRC is selected again directly after G40, only one of the two main axes is programmed.

```
%G239_Demo3.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20 (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
NO80 G40 (Deselect TRC without block)
(Y is not programmed again)
N090 G41 G01 X150 (Reselect TRC)
N100 G00 X200
N110 M30
```



Fig. 133: Motion block sequence of G239_only one main axis programmed

## Programing Example

## Example 4:

Here the 2nd motion block is only programmed in the 2nd motion block after TRC is reselected.

```
%G239 Demo4.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
NO30 V.G.WZ AKT.R = 10
N040 G41 G01 X30 Y20 (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40 (Deselect TRC without block)
N090 G41 G01 X150 (Reselect TRC)
N100 G00 X200 Y20
N110 M30
```



Fig. 134: Motion block sequence of G239_Program the 2nd main axis in 2nd motion block

## Programing Example

## Example 5:

In this example the position of second main axis is programmed as for the first selection (G41) after TRC is reselected after G40 .

```
%G239_Demo5.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239 (TRC mode)
N030 V.G.WZ AKT.R = 10
N040 G41 G01 X30 Y20 (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G01 X120
NO80 G40 (Deselect TRC without block)
N090 G41 G01 X150 Y20 (Reselect TRC)
N100 G00 X200
N110 G40
N120 M30
```



Fig. 135: Motion block sequence of G239_Program the 2 nd main axis as for the 1st selection

## Programing Example

## Example 6:

The arc programmed after TRC is deselected cannot be moved because the specified centre point $(M)$ no longer corresponds to the (new) starting point (S) and the (programmed) end point (E).

```
%G239_Demo_Kreis.nc
N005 G162
N010 G0 X0 Y0 Z0 F1000
NO20 G239 (TRC mode)
N030 V.G.WZ_AKT.R = 30
N040 G41 G01 X30 Y20 (Select TRC)
N050 G01 X60
N060 G01 X90
N070 G40 (Deselect TRC without block)
N080 G02 X140 Y-30 J-50 (Error, centre point not correct)
N100 G00 X200
N110 G40
N120 M30
```



Fig. 136: Motion block sequence of G239_Centre point does not match starting and end points

A message is output with ID 20035.

### 13.2.6 Direct selection/deselection of TRC on the path (G236)

2 motion blocks containing motion information are required to select or deselect G236.
The programmed motion blocks in the figures below run from S to E 1 and from E1 to E2.
It is permitted to select or deselect all combinations of linear and circular blocks.

## Notice

If the transition strategy G25 (Insert linear blocks) is used with G236, an arc is integrated at the corresponding transition angle when TRC is selected or deselected.
$\qquad$

## Selection

The tool radius is built up of the following:


Transition angle $0^{\circ}<\alpha<90^{\circ}$


Transition angle $90^{\circ}<=\alpha<=180^{\circ}$, TRC option G236_LIN [ 567$]=0$ (default)


Transition angle $90^{\circ}<=\alpha<=180^{\circ}$, TRC option G236_LIN [ $\left.>567\right]=1$


Transition angle $180^{\circ}<\alpha<=270^{\circ}$


Transition angle $270^{\circ}<\alpha<360^{\circ}$

## deselecting

In analogy to the tool radius composition, tool radius is reduced when TCR is deselected as follows:


Transition angle $0^{\circ}<\alpha<90^{\circ}$


Transition angle $90^{\circ}<=\alpha<=180^{\circ}$, TRC option G236_LIN [> 567]= 0 (default)


Transition angle $90^{\circ}<=\alpha<=180^{\circ}$, TRC option G236_LIN [ 567]= 1


Transition angle $180^{\circ}<\alpha<=270^{\circ}$


Transition angle $270^{\circ}<\alpha<360^{\circ}$

### 13.2.6.1 Selecting/deselecting $\mathbf{G} 236$ with closed contours

### 13.2.6.1.1 Selection/deselection at inside corners

Selection and deselection at an inside corner means that the transition angle $\beta$ between the first and last motion block of the contour is less than $180^{\circ}$.


Fig. 137: Selection and deselection of closed contours at inside corner

It is recommended to place the selection and deselection point of the TRC on the bisecting line in the selected corner to avoid damaging the contour.

The contour is damaged if the selection and deselection point is incorrectly placed.


Contour violation with G236 at inside corner

## Programing Example

```
%g236_test.nc
N01 G236 D1 F1000
NO2G01 X10 Y10 (Selection point)
N05 G42
N10 G01 X30 Y30
N20 G01 X30 Y20
;
N100 G01 X10 Y30
N105 G40
N110 G01 Y10 Y10 (Deselection point)
;
N999 M30
```


### 13.2.6.1.2 Selection/deselection at outside corners

Selection and deselection at an outside corner means that the transition angle $\beta$ between the first and last motion block of the contour is greater than $180^{\circ}$.

It is recommended to place the selection and deselection point of the TRC within the area marked green, as depicted in the fígure below.


Fig. 138: Selection and deselection of closed contours at outside corner

### 13.2.7 Generate compensation blocks

The generation of compensation blocks is not subject to restrictions with respect to the NC block sequence as applies to direct or indirect TRC selection. The next block after the current block is used to calculate contour transitions between the programmed NC blocks.
The contour transition is on straight lines (G25) by default - or optionally on arcs (G26) - with the option of feedrate adaptation (G10/G11).
The table below and the supplementary diagrams show all the possible contour transitions, whereby both the possible transitions (linear and circular interim blocks) are shown.

## Insert blocks by the TRC

LIN: Linear block, CIR: Circular block
G25: Insert linear transition
G26: Insert circular transition
(2 LIN : The TRC inserts 2 linear blocks at the transition).

|  | Angle range |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $0^{\circ}$ to $180^{\circ}$ |  | $180^{\circ}$ to $270^{\circ}$ |  | $270^{\circ}$ to $360^{\circ}$ |  |
| progr. block sequence | G25 | G26 | G25 | G26 | G25 | G26 |
| LIN-LIN | 0 | 0 | 0 | 1 CIR | 1 LIN | 1 CIR |
| LIN-CIR / CIR-LIN | 0 | 0 | 1 LIN | 1 CIR | 2 LIN | 1 CIR |
| CIR-CIR | 0 | 0 | 2 LIN | 1 CIR | 3 LIN | 1 CIR |



Fig. 139: Example of contour transition on straight lines for linear-linear block sequence


Fig. 140: Example of contour transition to an arc for linear-linear block sequence
Combination type $1: 0^{\circ}<\beta \leq 180^{\circ}$, G41

TRC transitions: Combination cases for $0^{\circ}<\beta \leq 180^{\circ}$ (independent of G25 or G26 since no compensation blocks are added)


TRC transitions: Combination cases for $180^{\circ}<\beta \leq 270^{\circ}$ with linear interim blocks


TRC transitions: Combination cases for $270^{\circ}<\beta \leq 360^{\circ}$ with linear interim blocks


TRC transitions: Combination cases for $180^{\circ}<\beta \leq 270^{\circ}$ with circular interim blocks
Combination type 1:270 $<\beta \leq 360^{\circ}, \mathrm{G} 41, \mathrm{G} 26$ Combination type 3: $270^{\circ}<\beta \leq 360^{\circ}, \mathrm{G} 41, \mathrm{G} 26$

TRC transitions: Combination cases for $270^{\circ}<\beta \leq 360^{\circ}$ with circular interim blocks

### 13.2.8 Reaction on contour change

A direct contour change, e.g. from G41 to G42, is permitted without any explicit deselection (G40) of TRC. The contour change corresponds to a TRC deselection and selection. Contour change should only be carried out for linear NC blocks for exit and approach blocks. Circular blocks are also permitted directly before and after a contour change.

## Notice

If a circular block is programmed during a contour change, an error message is output.

## Programing Example

Reaction on contour change

```
N05 G01 Y10 F100 G41 (Linear block with select. of TRC left)
V.G.WZR=2 (of the contour)
N10 X20 Y25
N20 X40 G42 (Linear block with select. of TRC right)
    (of the contour)
N30 X60 Y10
N40 X0 Y0 G40 (Deselect TRC)
N50 M30
```



Fig. 141: Example of a selection change without deselection

### 13.2.9 Reaction to change in tool radius

A change in tool radius is possible both within a linear block and within circular blocks.

## Programing Example

\%wr_lin.nc
N10 V.G.WZ AKT.R $=10$
N20 G00 X0 Y0 Z0 F1000
N25 G41 G1 X20 Y20
N30 G01 X100
N35 V.G.WZ AKT.R $=20$
N40 G01 X2000
N50 X240 Y100
N200 G40 X500
N999 M30


Fig. 142: Tool radius change within linear block

## Programing Example

## Change within a circular block

\%wr_rad.nc

N10 V.G.WZ AKT.R = 10
N20 G00 X0 Y0 Z0 F1000
N25 G41 G1 X20 Y20
N30 G01 X100
N35 V.G.WZ AKT.R $=20$
N40 G02 X180 Y100 R50
N50 X240 Y150

N200 G40 X500
N999 M30


Fig. 143: Tool radius change within circular block

### 13.2.10 Tangential selection/deselection of TRC (G05)

When TRC is directly selected/deselected, it generally results in a kink on the path contour at the start of machining. With angles greater than $180^{\circ}$, the contour of the block after selection and the block before deselection are violated.

Tangential entry and exit provides a solution to avoid these contour violations in direct selection/ deselection mode and to minimise the jerk that occurs at the kink points on the path.
G05 must be programmed in the same block in conjunction with G40, G41, G42. This then derives whether a tangential transition should take place at the start or at the end of the contour.

From the current position, the next contour element G01, G02... is approached tangentially in a circle at the programmed feedrate; where necessary, the feedrate must be adapted with G10/ G11.

G05 in conjunction with G41 or G42 causes tangential entry at contour start; G05 and G40 cause a tangential exit at contour end when G41/G42 is active. This converts the selection/deselection block into a circular block.

The motion blocks triggered by the G05 function when TRC is selected and deselected are illustrated on the next two pages. A total of four curves are shown resulting from different starting points (P1, P1') with the same programming.

## Selection in tangential mode:

The selection point (AnP) is calculated in the same as in conventional direct selection. The direction of rotation of the circular selection block is specified based on the orientation of the first selected block and the position of the starting point. The circle centre point (MP) results from the intersecting point of the centre perpendiculars from the starting point ( P 1 or $\mathrm{P} 1^{\prime}$ ) and the selection point (AnP) and the straight line starting point of the selection block (P2) - selection point (AnP).

## Deselection in tangential mode:

The last compensated end point (AbP), referred to as last selection point here, is calculated in the same way as in conventional direct deselection mode. The direction of rotation of the circular deselection block is specified based on the orientation of the last selected block and the position of the deselection point ( P 4 or P 4 '). The circular centre point results from the intersecting point of the centre perpendiculars to the connecting line of the last selection point (AbP) and the deselection point ( P 4 and $\mathrm{P} 4^{\prime}$ ) and the straight lines of the last selection point (AbP) - last programmed point (P3) before deselection.


Fig. 144: Selection and deselection of TRC in tangential mode


Fig. 145: Selection and deselection of TRC in tangential mode

### 13.2.11 <br> Adapting feed of TRC (G10/G11)

| G10 | Constant feedrate | (modal, initial state) |
| :--- | :--- | ---: |
| G11 | Adapted feedrate | (modal) |

When G11 is active, transition blocks inserted by tool radius compensation are travelled at a higher velocity. In addition, the programmed feedrate for circular blocks acts on the tool engagement point (see figure below).


Fig. 146: Adapting feedrate with compensated circular interpolation


### 13.2.12 Selecting/deselecting TRC contour masking (G140/G141)

| G140 | Deselecting contour masking | (modal, initial state) |
| :--- | :--- | :--- |
| G141 | Selecting contour masking | (modal) |

If a rough-machining operation with a large diameter tool is executed before finishing, there is a danger of contour violation if single contour elements have smaller dimensions than that of the tool geometry. In order to move along contiguous contour objects, it is possible to mask certain contour elements if violations are detected by using G140/G141 when TRC is active and then to process the following contour to avoid damaging it.
No error message is output if contour masking is still selected at program end. At program start the initial state G140 is active.

## Attention

Contour masking remains selected if tool radius compensation is deactivated by G40 and is re-activated when tool radius compensation is reselected.


Fig. 147: Masking of N20 to avoid contour violation.

### 13.2.13 <br> Limits of TRC

The TRC takes into account the current and two further relevant NC blocks to calculate interim blocks. "Relevant" here means: motion blocks in the selected main plane in which tool radius compensation takes place. "Non-relevant" would refer to techno blocks, time delays (G04) or path motions with one axis perpendicular to the main plane. If an interim block running in the opposite direction to the programmed contour is calculated within these three blocks, a contour violation is detected. The figure below shows an example of this.

If a bottleneck position is caused by one or more blocks which are more than 3 blocks away from current block, this contour violation is not detected by the TRC. The figure on the next page shows an example of this


Fig. 148: Example of detected contour violation


Fig. 149: Example of undetected contour violation

### 13.2.14 Programmable additional options of TRC (\#TRC)

The following command permits the programming of additional optional TRC functions.

Notice
After selection, these optional TRC functions remain active up to main program end or RESET, but they can also be deselected at any time while the NC program is running.

```
#TRC [ [ CONV_CIR_TO_LIN<expr> ] [ KERF_MASKING<expr> ] [ REVERSE<expr> ]
    [IGNORE_CONT_DAMAGE<expr>][REMOVE_MASKED_BLOCKS <expr> ]]
    [EXT_ANGLE_BLOCK_INTERSECTION<expr>]]
```

CONV_CIR_TO_LIN<expr> In circular blocks in which the tool radius is greater than the programmed radius of the circle element, this parameter converts the circular block directly into linear blocks. The effectiveness of this function is dependent on whether contour masking processes are active (G141).

| Value | Meaning |
| :--- | :--- |
| 0 | No conversion of circular blocks (default). |
| 1 | Conversion of circular blocks into linear blocks. |

KERF_MASKING<expr> This parameter explicitly selects the masking of kerfs.

| Value | Meaning |
| :--- | :--- |
| 0 | Kerf masking deactivated (default). |
| 1 | Kerf masking activated. |

## Notice

When contour masking is used, this TRC option is implicitly activated and/or deactivated.

The functionality of kerf masking is based on offsetting a programmed point as soon as the program detects that a kerf cannot be traversed by a tool.


- Corrected programmed contour

Fig. 150: Illustration of kerf masking
REVERSE<expr> This parameter permits the direct change (G41<->G42) of selection side with reversing motions when TRC is active.

| Value | Meaning |
| :--- | :--- |
| 0 | Direct change of selection side deactivated (default) |
| 1 | Direct change of selection side activated |

The change of selection side of TRC always takes place at the point of reversal. With linear blocks, this is dependent on exactly reversing motions.

With circular blocks, the tangents of both circles must be identical at the point of reversal and the directions of both circles must be different.

IGNORE_CONT_DAMAGE<expr> This parameter explicitly ignores contour violations.

| Value | Meaning |
| :--- | :--- |
| 0 | Contour violations not ignored (default) |
| 1 | Contour violations ignored |

REMOVE_MASKED_BLOCKS<expr>
This parameter deletes contour loops detected by contour masking. Pure motions of tracking axes are retained. From the viewpoint of TRC, this also includes a motion of the 3rd main axis. main axis.

The parameter is specially suited for contours with very short blocks. Effectiveness is dependent on whether contour masking is active (G141).

| Value | Meaning |
| :--- | :--- |
| 0 | Closed contour loops are not deleted (default). |
| 1 | Closed contour loops are deleted. |

EXT_ANGLE_BLOCK_INTERSECTION<expr>

This parameter changes the limit of the transition angle between two motion blocks from $180^{\circ}$ to $181^{\circ}$. This avoids the creation of additional TRC transition blocks in this angle range.
The value of the limit itself cannot be modified.

| Value | Meaning |
| :--- | :--- |
| 0 | Limit of transition angle is $180^{\circ}$ (default) |
| 1 | Limit of transition angle is $181^{\circ}$ |

## Programing Example

Conversion of circular blocks:

```
N1000 V.G.WZ_AKT.R=1 (Tool radius)
N2300 G140 (Deactivate contour masking))
N2450 #TRC [CONV_CIR_TO_LIN=1] (Activate option)
N2500 G41 (Select TRC left of contour)
(Circular element with radius less than tool radius)
N2550 G03 X3557.83 Y-577.61 I0.00 J0.60
(no direct conversion to a linear block)
N3000 G141 (Activate contour masking)
(Circular element with radius less than tool radius)
N3550 G03 X3557.83 Y-577.61 I0.00 J0.60
(direct conversion to a linear block)
N3600 G40
(Deselect TRC)
```


## Programing Example

```
N100 #TRC[REVERSE=1]
N110 G00 X0 Y0
N120 G01 X100 Y100
N130 G41 G01 X150 (Select TRC left of contour)
N140 G01 X250
N150 G01 Y200 (Reversal point)
N160 G42 (Change selection side, now right of contour)
N170 G01 X100
N180 G01 X100 Y150
N160 G40 G01 X0
N170 G01 Y0
```

This parameter defines whether a linear or a circular transition block is integrated when the selection/deselection method G236 [ 540] is active. This parameter only acts within the angle range between $90^{\circ}$ and $180^{\circ} \mathrm{C}$ when TRC is selected or deselected.

| Value | Meaning |
| :--- | :--- |
| 0 | A circular transition block is integrated at selection and deselection. (default) |
| 1 | A linear transition block is integrated at selection and deselection. |

### 13.2.14.1 TRC option STRETCH_FACTOR

STRETCH_FACTOR<expr> This parameter can influence the bypass strategy with acute outside corners. The parameter value determines the maximum block stretch. The length is dependent of the tool radius used.

The parameter only affects outside corners with a transition angle greater than 270 degrees.

| Value | Meaning |
| :--- | :--- |
| $1-9$ | Factor for possible block stretch <br> Default value: 1. |

## Notice

If the STRETCH_FACTOR is programmed as less than 1 or greater than 9 , a warning P-ERR-22007 is output and the default value is set.

Notice
With highly pointed outside corners and large STRETCH_FACTOR selected, block stretch may be very large.

Extreme block stretches are possible as well as possible overshoots of the software limit switch with corresponding error messages (P-ERR-120002/P-ERR-120003)

The option behaviour is illustrated by means of examples of block transitions linear-linear and cir-cular-circular. The function can also be used for linear-circular and circular-linear transitions.
\%stretch_test1.nc
N010 G139 G25
N020 V.G.WZ_AKT.R $=5$
(Use default value )
N040 \#TRC[STRETCH_FACTOR=1]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G1 X35 Y30
N120 G1 X30 Y20
N130 G1 Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30


If the STRETCH_FACTOR is further increased with the current geometries, the parallel path does not change. The corner point is used as with factor 3.

### 13.2.14.2 TRC option PERPENDICULAR_RADIUS_CHANGE

PERPENDICULAR_RADIUS_CHANGE<expr>

This parameter can extend the change perpendicularly if there is a change in tool radius.

| Value | Meaning |
| :--- | :--- |
| 0 | No perpendicular extension of the new tool radius (default) |
| 1 | Perpendicular extension of the new tool radius |

In the program examples, a change is made to the tool radius between the programmed motion blocks N50 and N60.
To illustrate the behaviour, a very large change in tool radius is chosen. Normally, a change in tool radius relate to very small corrections.
Several block transitions are shown in the examples below. All combinations of linear and circular blocks are permitted.

Change of tool radius at inside corner

|  | ```\%Test.nc N05 G0 X0 Y0 F1000 \#TRC [PERPENDICULAR_RA- DIUS_CHANGE=1] N10 V.G.WZ_AKT.R= 5 N20 G42 N40 G01 X20 Y0 N50 G01 X50 Y50 N55 V.G.WZ_AKT.R = 6 N60 G01 X100 N70 G40 X120 Y0 N99 M30``` |
| :---: | :---: |
|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1 ] <br> N10 V.G.WZ_AKT.R=6 <br> N20 G42 <br> N40 G01 X20 Y0 <br> N50 G01 X50 Y50 <br> N55 V.G.WZ_AKT.R = 5 <br> N60 G01 X100 <br> N70 G40 X120 Y0 <br> N99 M30 |
| Workpiece | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1 ] <br> N10 V.G.WZ_AKT.R= 5 <br> N20 G42 <br> N30 G01 X10 Y20 <br> N40 G01 X20 <br> N50 G03 X50 Y25 R30 <br> N55 V.G.WZ_AKT.R = 6 <br> N60 G03 X90 Y18 R40 <br> N70 G01 X100 <br> N80 G40 X120 Y0 <br> N99 M30 |
| Workpiece | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1] <br> N10 V.G.WZ_AKT.R=6 <br> N20 G42 <br> N30 G01 X10 Y20 <br> N40 G01 X20 <br> N50 G03 X50 Y25 R30 <br> N55 V.G.WZ_AKT.R = 5 <br> N60 G03 X90 Y18 R40 <br> N70 G01 X100 <br> N80 G40 X120 Y0 <br> N99 M30 |

Change of tool radius at outside corner

|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1 ] <br> N10 V.G.WZ_AKT.R= 6 <br> N20 G42 G25 <br> N40 G01 X20 Y20 <br> N50 G01 X50 <br> N55 V.G.WZ_AKT.R = 5 <br> N60 G01 X80 Y50 <br> N70 G01 X120 <br> N80 G40 X140 Y0 <br> N99 M30 |
| :---: | :---: |
|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1 ] <br> N10 V.G.WZ_AKT.R=6 <br> N20 G42 G26 <br> N40 G01 X20 Y20 <br> N50 G01 X50 <br> N55 V.G.WZ_AKT.R = 5 <br> N60 G01 X80 Y50 <br> N70 G01 X120 <br> N80 G40 X140 Y0 <br> N99 M30 |
|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1] <br> N10 V.G.WZ_AKT.R=5 <br> N20 G42 G25 <br> N40 G01 X20 Y20 <br> N50 G01 X50 <br> N55 V.G.WZ_AKT.R = 6 <br> N60 G01 X80 Y50 <br> N70 G01 X120 <br> N80 G40 X140 Y0 <br> N99 M30 |
|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1 ] <br> N10 V.G.WZ_AKT.R=5 <br> N20 G25 <br> N30 G42 X10 Y10 <br> N40 G01 X20 <br> N50 G02 X60 Y0 R30 <br> N55 V.G.WZ_AKT.R = 6 <br> N60 G02 X90 Y12 R50 <br> N70 G01 X100 <br> N80 G40 X120 Y0 <br> N99 M30 |


|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1] <br> N10 V.G.WZ_AKT.R=5 <br> N20 G26 <br> N30 G42 X10 Y10 <br> N40 G01 X20 <br> N50 G02 X60 Y0 R30 <br> N55 V.G.WZ_AKT.R = 6 <br> N60 G02 X90 Y12 R50 <br> N70 G01 X100 <br> N80 G40 X120 Y0 <br> N99 M30 |
| :---: | :---: |
|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1] <br> N10 V.G.WZ_AKT.R=5 <br> N20 G25 <br> N30 G42 X10 Y10 <br> N40 G01 X20 <br> N50 G02 X60 Y0 R30 <br> N55 V.G.WZ_AKT.R = 6 <br> N60 G01X80 Y40 <br> N70 G01 X100 <br> N80 G40 X120 Y0 <br> N99 M30 |
|  | \%Test.nc <br> N05 G0 X0 Y0 F1000 <br> \#TRC [PERPENDICULAR_RA- <br> DIUS_CHANGE=1] <br> N10 V.G.WZ_AKT.R=5 <br> N20 G26 <br> N30 G42 X10 Y10 <br> N40 G01 X20 <br> N50 G02 X60 Y0 R30 <br> N55 V.G.WZ_AKT.R = 6 <br> N60 G01X80 Y40 <br> N70 G01 X100 <br> N80 G40 X120 Y0 <br> N99 M30 |

### 13.2.15 Exception list of commands with active TRC/SRK

Below is a list of commands which are not permitted when TRC/SRK (G41/G42) are active:
(This refers to the section in the NC program between G41/G42 and G40.)

## Notice

Error 20651 is output between G41/G42 and G40 if they are followed by commands.

## Notice

When the TRC modes G139/G138/G236/G05 are used, an additional motion block is required after G 40 to reduce the tool radius.
If subsequent commands are used directly after G40, error 90050 is output.

| G command | Note |
| :--- | :--- |
| G200/ G201 |  |
| G303 | Circles in space |

A read access by synchronous V.E. variables is also not permitted.

| Additional command | Note |
| :--- | :--- |
| \#AX DEF |  |
| \#AX DEF DEFAULT |  |
| \#AX LINK ON/OFF/OFF ALL |  |
| \#AX REQUEST | Release axes. <br> Axes that may not be released in the active <br> plane. |
| \#AX RELEASE |  |
| \#AX RELEASE ALL |  |
|  |  |
| \#CALL AX |  |
| \#CAX |  |
| \#CAX OFF |  |
| \#CHANNEL INIT |  |
| \#CLEAR CONFIG |  |
| \#CS ON/OFF |  |
| \#CS MODE ON/OFF |  |
| \#CYL |  |
| \#CYL OFF |  |
| \#CYL ORI LATERAL |  |
| \#CYL ORI PROFILE |  |
|  | \#FACE |


| \#FACE OFF |  |
| :--- | :--- |
|  |  |
| \#FLUSH | Output is shifted to CNC but is permissible. |
| \#FLUSH CONTINUE |  |
| \#FLUSH WAIT |  |
|  |  |
| \#GET CMDPOS |  |
| \#GET ACTPOS |  |
| \#GET MANUAL OFFSETS |  |
|  |  |
| \#LOAD CONFIG |  |
|  |  |
| \#MCS ON/OFF |  |
| \#MCS TO WCS |  |
|  | Release axes. <br> Axes that are not in the active place may not <br> be released. |
| \#OTC OFF |  |
|  |  |
| \#PRESET |  |
| \#PTP ON/OFF |  |
| \#PUT AX |  |
| \#PUT AX ALL |  |
|  | \#SET AX |
| \#SET AX LINK |  |
|  |  |
| \#TRACK CS ABS/ON/OFF |  |
|  |  |
|  |  |

## 14 Variables and calculation of variables

A complete list of programming variables is contained in the overview of commands in Variable programming (V.) [> 816].

Here, variables mean on the one hand an internal datum of the decoder with fixed name assignment; on the other hand they refer to self-defined variables whose designation is essentially freely selectable. With the exception of external variables (V.E...), their validity and utilisation are limited exclusively to a particular NC channel.

```
General syntax:
V.<NAME_1>.<NAME_2>.<NAME_3>.{<NAME_n>.}
```

| V. | indicates access to the variables |  |
| :---: | :---: | :---: |
| <NAME_1>. | Defines the global data designation: |  |
|  | "A." | represents axis-specific variables |
|  | $\begin{aligned} & \text { "SPDL." } \\ & \text { "SPDL_PRO } \\ & \text { G." } \end{aligned}$ | represents spindle-specific variables |
|  | "G." | represents inter-axis variables globally valid in the channel |
|  | "E." | represents external variables |
|  | "P." | represents self-defined variables valid up to the end of the main program (M2, M30), |
|  | "S." | represents self-defined variables valid throughout the main program. |
|  | "L." | represents self-defined local variables that are valid until the current program level is left by return (M17, M29), |
|  | "CYC." | represents variables that may only be used in cycle programs (L CYCLE). |
| <NAME_2>. | specifies the data name |  |
| <NAME_3>. | e.g. indicates the index if a distinction is made between several identical data. |  |

## Notice

## Programming axis identification

The last identification code represents the axis code of axis-specific and several group-specific variables. Here, the designations

$$
\begin{aligned}
& \text { ".X" or "[0]" } \\
& \text { ".Y" or "[1]" } \\
& \text { ".Z" or "[2]" }
\end{aligned}
$$

must be selectively used if the name " X " is assigned to the axis with the index 0 , the name " $Y$ " is assigned to the axis with the index 1 and the name " $Z$ " is assigned to the axis with index 2 in the channel parameter list.

## Example

## Absolute value of the $X$ axis:

V.A.ABS.X or V.A.ABS[0]

Analogous to spindle-specific variables, the spindle names or the corresponding indices must be used as listed in the channel parameter synchronisation mode.

$$
\begin{aligned}
& \text { ".S" or "[0]" } \\
& \text { ".S2" or "[1]" etc. }
\end{aligned}
$$

## Example

## Logical axis number of the $S$ spindle:

Logical axis number of the $\mathbf{S}$ spindle:

> V.SPDL.LOG_AX_NR.S or V.SPDL.LOG_AX_NR[0]

## Programing Example

The lines N20/N30 cause a linear interpolation in $X$ direction by the value of the variables V.A.BZP.Y, i.e. the reference point offset in $Y$ direction.

```
N10 G92 X0 Y40 Z0 ;Reference point offset
N20 G91 G01 F1000 XV.A.BZP.Y
N30 XV.A.BZP[1] ;here: axis index 1 == Y
N40 M30
```

The content can be read by all variables and a value can also be assigned to several of them. The access type is firmly given for each variable, however generally only a reading access is allowed. Because for most of the variables a writing access is not practical.

## Programing Example



Here, the 2 nd zero offset vector for the axis with the index 1 is assigned the value 100 :

```
N10 V.G.NP[2].V[1] = 100
```

The EXIST function (see Section Arithmetic expressions <expr> [ 31]) checks whether a variable exists at all.

## Programing Example

The EXIST request for an axis-specific variable checks whether a specific axis is found at all in the NC channel.

```
N10 G90 Y0
N20 $IF EXIST[V.A.LOG_AX_NR.X] == TRUE
N30 X-10 ;X axis is in channel, approach position -10
N40 $ELSE
N50 #CALL AX [X,1,0] ;X axis not in channel, request first
N60 $ENDIF
```

M30

Before access to an external variable, a check is made whether access is possible at all:

```
N10 G90 Y0
N20 $IF EXIST[V.E.POS_1] == TRUE
N30 XV.E.POS_1 ;Move X axis to position POS_1
N40 $ELSE
N50 #MSG ["V.E.POS_1 not found!"] ;message is output.
N55 M0 ;Stop
N60 $ENDIF
M30
```


### 14.1 Axis-specific variables (V.A.)

The code for axis-specific variables is "V.A. ...".
The axis-specific identifier can be specified in 2 variants:
1: Axis name according to channel list (represented with " $X$ " as example below)
2: Axis index [i] according to channel list where <i>: $0 . . .31$
Example:
V.A.ABS.X or V.A.ABS[0]

## Attention

V.A. variables can only be programmed for linear and rotary axes but not for spindles.

| V.A.<var_name> | Meaning | Data type | Unit for input/output | Permitted access <br> Read/ <br> Write |
| :---: | :---: | :---: | :---: | :---: |
| MENT. X | Virtual coordinate of the previous NC block (see Section Mirroring G20-G23 [ 114]) | Real | [mm, inch] | R |
| PROG.X | Programmed coordinate of the previous NC block During active contour rotation (\#ROTATION) the variable supplies the coordinate value mapped onto the machine axes. | Real | [mm, inch] | R |
| ABS. X | Absolute coordinate of the previous NC block or current absolute coordinate after NC command \#CHANNEL INIT [> 179] each in the currently active coordinate system | Real | [mm, inch] | R |
| ACS.ABS.X | Current actual axis position in the present coordinate system without offsets. | Real | [mm, inch] | R |
| ACS.ABS.X | Current actual position during active transformation mapped onto the machine axes. | Real | [mm, inch] | R |
| -SWE.X | Current effective negative software limit switch | Real | [mm, inch] | R |
| +SWE.X | Current effective positive software limit switch | Real | [mm, inch] | R |
| -SWE_MDS.X | Configured negative software limit switch (acc. to P-AXIS-00177). | Real | [mm, inch] | R |
| +SWE_MDS.X | Configured positive software limit switch (acc. to P-AXIS-00178). | Real | [mm, inch] | R |
| REF.X | Machine reference point (only assigned after successful machine reference search) | Real | [mm, inch] | R |
| BZP. X | Reference point offset | Real | [mm, inch] | R |
| PZV.X | Clamp position offset | Real | [mm, inch] | R |


| MESS.X | After measurement run completed, supplies <br> the axis-specific measured value in the co- <br> ordinate system in which measurement took <br> place. The value includes all offsets in the <br> calculation <br> With 2.5D: ACS values or with CS / <br> TRAFO: PCS values | $R$ | [mm, inch] |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- |



## Release Note

As of Build V2.11.2020.07, the axis-specific variables V.A.MEAS.ACS.VALUE and V.A.MEAS.PCS.VALUE supplement the variable V.A.MESS. The additional variables supply the measured value both in the axis coordinate system including all offsets and also the measured value in the programming coordinate system without offsets.

| MEAS.ACS.VALUE. X | After measurement run completed, supplies the axis-specific measured value in the axis coordinate system (ACS). The value contains all offsets. | Real | [mm, inch] | R |
| :---: | :---: | :---: | :---: | :---: |
| MEAS.PCS.VALUE. X | After measurement run completed, supplies the axis-specific measured value in the programming coordinate system (PCS). The value does not contain any offsets | Real | [mm, inch] | R |
| MOFFS.X | Measurement offset | Real | [mm, inch] | R |
| MERF.X | Measurement run completed? If yes, then 1 | Boolea <br> n | 0,1 | R |
| MEIN.X | Included measuring offset | Real | [mm, inch] | R |
| RERF.X | Homing completed? If yes, then 1 | Boolea <br> n | 0,1 | R |
| MANUAL_OFFSETS.X or SOFFS.X | Motion path in manual mode. Only practical in conjunction with NC command \#GET MANUAL OFFSETS [ 177]. | Real | [mm, inch] | R |
| MODE.X | Current axis mode | Integer | - | R |
| MODULO_VALUE.X | Modulo range | Real | [ ${ }^{\circ}$ ] | R |
| LOG_AX_NR.X | Logical axis number of an axis | Integer | - | R |
| AX_LIST_NAME.X | Configured axis name (acc. to P-AXIS-00297) | String | - | R |
| AXIS_DEACTIVATED. $X$ | Variable indicates whether the axis was parked via the HLI. <br> Available as of V2.11.2813 | Boolea <br> n | - | R |
| EN- <br> CODER2 VALUE.X | Current value of a second encoder (optional) in the drive | Real | [mm, inch] | R |
| MIRROR.X | Mirror status of axis <br> (1: no mirroring -1: mirroring) | Integer | - | R |


| wcs.x MCS.X | Convert between machine coordinates (MCS) and workpiece coordinates (WCS). Only practical in conjunction with the NC commands \#WCS TO MCS [〉 744] and \#MCS TO WCS [ 744] | Real | [mm, inch] | R/W |
| :---: | :---: | :---: | :---: | :---: |
| DIA- <br> METER_PROG.ABS. X | Supplies the value of P-AXIS-00058 if the following applies: <br> - Diameter programming (G51) is active <br> - Axis is configured as face turning axis <br> [as of CNC Build V2.11.2051.00] | Boolea <br> n | 0,1 | R |
| DIA- <br> METER_PROG.REL. <br> X | Supplies the value of P-AXIS-00059 if the following applies: <br> - Diameter programming (G51) is active <br> - Axis is configured as face turning axis <br> [as of CNC Build V2.11.2051.00] | Boolea <br> n | 0,1 | R |


| CROSS_COMP_INIT.X | Is cross compensation initialised for the axis? <br> If yes, then 1 | Boolean | 0,1 | $R$ |
| :--- | :--- | :--- | :--- | :--- |
| PLANE_COMP_INIT.X | Is plane compensation initialised for the axis? <br> If yes, then 1 | Boolean | 0,1 | R |
| LEAD_COMP_INIT.X | Is leadscrew error compensation activated for <br> the axis? If yes, then 1 | Boolean | 0,1 | R |
| TEMP_COMP_INIT.X | Is temperature compensation initialised for <br> the axis? If yes, then 1 | Boolean | 0,1 | R |
| CROSS_COMP_ACTIVE.X | Is cross compensation active for the axis? If <br> yes, then 1 | Boolean | 0,1 | R |
| PLANE_COMP_ACTIVE.X | Is plane compensation active for the axis? If <br> yes, then 1 | Boolean | 0,1 | R |
| LEAD_COMP_ACTIVE.X | Is leadscrew error compensation active for <br> the axis? If yes, then 1 | Boolean | 0,1 | R |
| TEMP_COMP_ACTIVE.X | Is temperature compensation active for the <br> axis? If yes, then 1 | Boolean | 0,1 | R |

As of CNC Build V2.11.2810 the following V.A. variables of the current compensation values are available.

| LEAD_COMP_CURR.X | Current compensation value of LSEC for the <br> axis | Real | [mm, in <br> ch] | $R$ |
| :--- | :--- | :--- | :--- | :--- |
| ROSS_COMP_CURR.X | Current compensation value of cross com- <br> pensation for the axis | Real | [mm, in <br> ch] | $R$ |
| PLANE_COMP_CURR.X | Current compensation value of plane com- <br> pensation for the axis | Real | [mm, in <br> ch] | $R$ |
| TEMP_COMP_CURR.X | Current compensation value of temperature <br> compensation for the axis | Real | [mm, in <br> ch] | $R$ |

When strings are used for axis designation (e.g. X_SLIDE, see also Sec. 3.1 [ $>44$ ]) these axis names are used to identify the variables.

Example: V.A.MENT.X_SCHLITTEN

## Programing Example



### 14.2 Spindle-specific variables (V.SPDL., V.SPDL_PROG.)

The identifier for variables that permit access to configuration-specific spindle data is "V.SPDL. ...".
The identifier for spindle data assigned by programming is "V.SPDL_PROG. ...".

| V.SPDL.<name> | Meaning | Data type | Unit for in-put/output | Permitted access Read / Write |
| :---: | :---: | :---: | :---: | :---: |
| LOG_AX_NR.S | Logical axis number of the spindle | Integer | - | R |
| PLC_CONTROL.S | Is spindle a PLC spindle? If yes, then 1 | Boolean | 0,1 | R |
| NBR_IN_CHANNEL | Total number of available spindles in the current NC channel | Integer | - | R |
| M_FCT_FREE | What is the classification of the M functions M3, M4, M5, M19? <br> Explicitly defined as spindle M functions: 0 <br> Free available for other techno functions: 1 | Boolean | 0,1 | R/W |


| V.SPDL_PROG.<name> | Meaning | Data type | Unit for in-put/output | Permitted access Read/ Write |
| :---: | :---: | :---: | :---: | :---: |
| SPEED.S | Current speed S.. of spindle | Real | [rpm] | R |
| MOVE_CMD.S | Current motion type of spindle: For M3: 3 , for M4: 4 , for M5: 5 | Integer | - | R |
| POSITION.S | Current set position S.POS.. of spindle for M19 | Integer | [ ${ }^{\circ}$ | R |
| MAX_SPEED.S | Maximum speed G196 S.. of spindle with G96. Only for main spindles. | Real | [rpm] | R |
| CONST_CUT_SPEED.S | Constant cutting speed G96 S.. for turning work. Only for main spindles. | Real | [m/min, $\left.\mathrm{ft} / \mathrm{min}{ }^{*}\right]$ | R |
| DWELL_ROT_COUNT.S | Dwell time G04 S.. in number of spindle revolutions. <br> Only for main spindles. | Real | [rpm] | R |
| GEAR_DATA_STEP.S | Currently set gear stage G112 S.. Only for main spindles. | Integer | - | R |

* [as of Build V2.11.2032.08 with G70 and P-CHAN-00360 = 1]


## Attention

Write access causes permanent change to internal channel parameters (P-CHAN-00098)
kernel Industrielle Steuerungstechnik GmbH

## Programing Example

A check is first made before programming a spindle whether it is known in the channel:
...
N10 G90 Y0
N20 \$IF EXIST[V.SPDL.LOG AX NR.S] == TRUE
N30 M3 S1000 (Spindle $S$ at speed 1000 rpm)
N40 \$ELSE
N50 \#MSG ["Spindle $S$ is not present!"] (Output message and stop)
N55 M0
N60 \$ENDIF
. . .
M30

### 14.3 Global variables (V.G.)

The identifier for inter-axis globally valid variables in the channel is "V.G. ...".
The axis-specific identifier can be specified in 2 variants:
1: Axis name according to channel list (represented with " $X$ " as example below)
2: Axis index [i] according to channel list where <i>: $0 . . .31$
Example:
V.G.NP_AKT.V.X or V.G.NP_AKT.V[0]

| V.G.<var_name> | Meaning | Data type | Unit for input/ output | Permitted access: Read/ Write |
| :---: | :---: | :---: | :---: | :---: |
| BLOCK_NR | The last programmed NC block number | Integer | - | R |
| MASS_MM | If measuring unit [mm], then 0 | Boolean | 0,1 | R |
| MASS_360 | If measuring unit [ ${ }^{\circ}$ ], then 0 | Boolean | 0,1 | R |
| I | I coordinate of circular programming. With active centre point compensation (G165), access to corrected value | Real | [mm, inch ] | R |
| J | $J$ coordinate of circular programming. With active centre point compensation (G165), access to corrected value | Real | [mm, inch ] | R |
| K | K coordinate of circular programming. With active centre point compensation (G165), access to corrected value | Real | [mm, inch ] | R |
| R | The radius traversed during circular interpolation | Real | [mm, inch ] | R |
| FEEDRATE | The last programmed feedrate (F word) | Real | [mm/min, $\mathrm{m} / \mathrm{min}$, inch/min] | R |
| FEEDRATE_MODE | G number of current feedrate mode: <br> 93: G93 active, 94: G94 active, 95: G95 active, 194: G194 active [as of V2.11.2039.01] | Integer | - | R |
| FEEDRATE_SCALE | Permits adaptation of a feedrate programmed in mm/ min to the unit specified in 'prog_start.feedrate_factor' (P-CHAN-00108). <br> Supplies the value 1000 for 'prog_start.feedrate_factor' $=0.1$ ( $\mathrm{m} / \mathrm{min}$ ) 1 for 'prog_start.feedrate_factor' $=100(\mathrm{~mm} / \mathrm{min})$ Programming example: <br> F2000 / V.G.FEEDRATE_SCALE results for 1000: F2 ( $\mathrm{m} / \mathrm{min}$ ) <br> 1: F2000 ( $\mathrm{mm} / \mathrm{min}$ ) [as of V2.11.2024.00] | Integer | - | R |
| MERR[i] | Compensation offset of circle centre point in the main axes of the current plane (<i> $=0,1$ ) | Real | $\begin{aligned} & \text { [mm, inch } \\ & ] \end{aligned}$ | R |

The "WZ[j]" variables permit read access to the data of any tool. They are both available with an external tool management system (transparent access) and for use of an internal tool table (<j> then corresponds to the index of the tool (or the tool number) in the tool list [5] [ 819]).

Write access is only permitted if an internal tool table is used.

| WZ[j].R | Radius of the tool | Real | [mm, inc h] | R/W* |
| :---: | :---: | :---: | :---: | :---: |
| WZ[j].L | Length of the tool | Real | [mm, inc h] | R/W* |
| WZ[j].P[i] | Tool parameters | Real | - | R/W* |
| WZ[j].V[i] or WZ[j].V.X | Offset in axis <i> or "X" of the tool | Real | [mm, inc h] | R/W* |
| WZ[j].ME | Measuring unit of radius, length and axis offsets, always supplies 0 (for [mm]) when a tool list is used, otherwise the variable has no significance | Boolean | 0,1 | R |
| WZ[j].OK | Validity flag of the tool; if valid, then 1 | Boolean | 0,1 | R/W* |
| WZ[j].SPDL_AX_NR | Logical axis number of the assigned spindle | Integer | - | R/W* |
| WZ[j].KIN_PARAM[i] | Kinematic parameters of the tool in internal unit | Real | $\begin{aligned} & {[0.1 \mu \mathrm{~m},} \\ & \left.10^{-40}\right] \end{aligned}$ | R/W* |
| WZ[j].KIN_ID | Kinematics ID of the tool | Integer | - | R/W* |
| WZ[j].TYPE | Tool type (0: Milling tool 1: Turning tool 2: Grinding tool) | Integer | - | R |
| WZ[j].TOOL_FIXED | Tool is alignable or fixed | Boolean | 0,1 | R/W* |
| WZ[j].SRK_ID | Cutter orientation of a turning tool | Integer | - | R/W* |
| WZ[j].S_MIN_SPEED | Minimum rotational speed (tool dynamic data) | Real | [rpm] | R/W* |
| WZ[j].S_MAX_SPEED | Maximum rotational speed (tool dynamic data) | Real | [rpm] | R/W* |
| WZ[j].S_MAX_ACC | Maximum acceleration (tool dynamic data) | Real | $\left[{ }^{\circ} / s^{2}\right]$ | R/W* |
| WZ[j].SISTER_VALID | Validity flag of sister tool (TOOL ID) | Boolean | 0,1 | R/W* |
| WZ[j].SISTER | Number of valid sister tool | Integer | - | R/W* |
| WZ[j].VARIANT_VALID | Validity flag of variant tool (TOOL ID) | Boolean | 0, 1 | R/W* |
| WZ[j].VARIANT | Number of valid variant tool | Integer | - | R/W* |
| WZ[j].GOBJECT[i].* | Access to the subelements of a specific graphic object (see FCT-C15) [as of V3.01.3018.00] | - | - | $\ldots$ |
| WZ[j].LINKPOINT.* | Access to the subelements of associated link points (see FCT-C15) [as of V3.01.3018.00] | - | - | ... |

$S^{*}$ : Write access to data of the internal tool management system as of CNC Build V3.1.3079.08

The variables "WZ_AKT", "T_AKT" and "D_AKT" permit access to the data of the currently selected tool. These variables are available both for an external tool management system and for the use of an internal tool table.

| T_AKT | Number of the selected tool | Integer | - | R |
| :--- | :--- | :--- | :--- | :--- |
| D_AKT | Number of the selected tool compensation record | Integer | - | $R$ |

## Notice

A write access always causes the temporary change of tool data as long as this function is selected. When a new tool is selected ( $D x x$ ) or deselected (D0), the changed data are lost.

## Exception:

When an external tool management system is used, the so-called additional tool data (V.G.WZ_AKT.P[i]) is adopted and saved when a new tool is selected or a tool is deselected (P-CHAN-00103).

| WZ_AKT.R | Radius of the selected tool | Real | [mm, inc h] | R/W |
| :---: | :---: | :---: | :---: | :---: |
| WZ_AKT.L | Length of the selected tool | Real | [mm, inc h] | R/W |
| WZ_AKT.P[i] | Free parameters of the selected tool where <i>: $0 . .59$ | Real | - | R/W |
| WZ AKT.V[i] or WZ_AKT.V.X | Offset in axis <i> or "X" of the selected tool | Real | [mm, inc h] | R/W |
| WZ_AKT.ME | Measuring unit of radius, length and axis offsets of the selected tool, always supplies 0 (for [mm]) when a tool list is used, otherwise the variable has no significance | Boolean | 0,1 | R |
| WZ_AKT.OK | Validity flag of the selected tool; is always 1 since only data of valid tools are adopted. An error message is output if invalid tools are requested | Boolean | 0,1 | R/W* |
| $\begin{aligned} & \text { WZ_AKT.SPDL_AX_N } \\ & \text { R } \end{aligned}$ | Logical axis number of the assigned spindle | Integer | - | R/W* |
| WZ_AKT.KIN_PARAM[ i] | CAUTION: Note on write access: Value must be programmed in internal units. <br> Kinematic parameters of the selected tool where <i>: 0 ... | Real | $\begin{aligned} & {[0.1 \mu \mathrm{~m},} \\ & \left.10^{-40^{\circ}}\right] \end{aligned}$ | R/W |
| WZ_AKT.KIN_ID | Kinematic ID of the selected tool | Integer | - | R/W* |
| WZ_AKT.TYPE | Tool type of the selected tool ( 0 : Milling tool 1 : Turning tool 2 : Grinding tool) | Integer | - | R |
| $\begin{aligned} & \text { WZ_AKT.TOOL_FIXE } \\ & \text { D } \end{aligned}$ | Tool is alignable or fixed | Boolean | 0,1 | R/W |
| WZ_AKT.SRK_ID | Cutter orientation of the selected turning tool | Integer | - | R/W* |
| WZ_AKT.S_MIN_SPE ED | Minimum rotational speed (tool dynamic data) | Real | [rpm] | R/W* |
| $\begin{aligned} & \text { WZ_AKT.S_MAX_SPE } \\ & \text { ED } \end{aligned}$ | Maximum rotational speed (tool dynamic data) | Real | [rpm] | R/W* |
| WZ_AKT.S_MAX_ACC | Maximum acceleration (tool dynamic data) | Real | [ ${ }^{\circ} \mathrm{s}^{2}$ ] | R/W* |
| WZ AKT.SISTER_VALID | Validity flag of sister tool (TOOL ID) | Boolean | 0,1 | R/W* |
| WZ_AKT.SISTER | Number of valid sister tool | Integer | - | R/W* |
| WZ AKT.VARIANT_VALID | Validity flag of variant tool (TOOL ID) | Boolean | 0,1 | R/W* |
| WZ_AKT.VARIANT | Number of valid variant tool | Integer | - | R/W* |
| WZ_AKT.WEAR_RADIUS | Total radius wear with radius compensation (OTC) (sum of discrete + continuous wear) | Real | [mm, inc h] | R |
| WZ_AKT.WEAR_RADIUS_CONT | Continuous radius wear with radius compensation (OTC) | Real | [mm, inc h] | R |
| WZ_AKT.WEAR[i] or WZ_AKT.WEAR.X | Wear in axis <i> or "X" with length compensation (OTC) | Real | [mm, inc h] | R |


| WZ_AKT.WEAR_CON <br> ST | Wear constant (OTC) | Real | $[0.1 \mu \mathrm{~m} /$ <br> $\mathrm{m}]$ | R/W |
| :--- | :--- | :--- | :--- | :--- |
| WZ_AKT.GOB- <br> JECT[i].* | Access to the subelements of a specific graphic object <br> (see FCT-C15) [as of V3.01.3018.00] | - | - | - |
| WZ_AKT.LINKPOINT.* | Access to the subelements of associated link points <br> (see FCT-C15) [as of V3.01.3018.00] | - | - | - |

$S^{*}$ : Write access to these current tool data as of CNC Build V3.1.3079.08

| NP[j].V[i] or <br> NP[j].V.X | Zero offset of an axis < i> or "X" <br> CAUTION: Write access causes permanent changes to <br> internal zero offset data. | Real | [mm, inch] | R/ <br> W |
| :--- | :--- | :--- | :--- | :--- |
| NP[j].ALL | Address all axes of a zero offset group. <br> CAUTION: Write access causes permanent changes to <br> internal zero offset data. | Real | [mm, inch] | R/ <br> W |
| NP_AKT.V[i] or <br> NP_AKT.V.X | Currently active zero offset of an axis <i> or "X" | Real | [mm, inch] | R/ <br> W |
| NP_AKT.ALL | Address currently active zero offsets of all axes | Real | [mm, inch] | R/ <br> W/ |
| NP_AKT.IDX | Index of the (currently active) zero offset group (e.g. 0 <br> with G53, 1 with G54, etc.) | Integer | - | R |
| NP_DEFAULT | Index of zero offset group after start-up effective in ini- <br> tial state | Integer | - | R/ <br> W |


| CNC_CHANNEL | Channel number | Integer | - | R |
| :---: | :---: | :---: | :---: | :---: |
| IPO_COUNT | System time counter | Integer | - | R |
| CS_ACTIVE | CS (\#CS...) active? If active, then TRUE | Boolean | 0,1 | R |
| CS_COUNT | Number of active (linked) CS | Integer | - | R |
| ACS_ACTIVE | ACS (\#ACS...) active? If active, then TRUE | Boolean | 0,1 | R |
| ACS_COUNT | Number of active (linked) ACS | Integer | - | R |
| CS | Active tool compensation (1: RTCP 2: TLC 3: Fixture adaptive coordinate system (ACS) on/off | Integer | - | R |
| TOOL_COMP | Current clamp position offset index | Integer | - | R |
| AX_LINK.NR | Number of current or last active coupling group | Integer | - | R |
| AX_LINK.ACTIVE | Is an axis coupling (\#AX LINK ON) active? If active, then 1 | Boolean | 0,1 | R |
| AX_LINK_GROUP[i].A CTIVE | Is a specific coupling group with the number < i> active? If active, then 1 | Boolean | 0,1 | R |
| ROT_ACTIVE | Contour rotation (\#ROTATION...) active? If active, then 1 | Boolean | 0,1 | R |
| ROT_ANGLE | Angle of contour rotation | Real | [ ${ }^{\circ}$ ] | R |
| ROT_CENTER1 | Offset of 1st main axis to rotation point with contour rotation | Real | [mm, inch] | R |
| ROT_CENTER2 | Offset of 2nd main axis to rotation point with contour rotation | Real | [mm, inch] | R |
| TIMER[i] | Counter value of timer with number i | Integer | [ms] | R |
| PROG_ABS | Measuring system, 0: G91 active 1: G90 active | Boolean | 0,1 | R |
| ACT_PLANE | G number of current active plane: <br> 17: G17 active, 18: G18 active, 19: G19 active | Integer | - | R |
| CNC_RELEASE | Build number of CNC version (old syntax) | Integer | - | R |
| CNC_VERSION.MAJOR | Major version of CNC [as of V2.11.2800] (e.g. 2 with V2.11.2807.42) | Integer | - | R |
| CNC_VERSION.MINOR | Minor version of CNC [as of V2.11.2800] (e.g. 11 with V2.11.2807.42)) | Integer | - | R |
| CNC_VERSION.BUILD | Build number of CNC version [as of V2.11.2800] (e.g. 2807 with V2.11.2807.42)) | Integer | - | R |
| CNC_VERSION.PATCH | Patch number of CNC version [as of V2.11.2800] (e.g.. 42 with V2.11.2807.42) | Integer | - | R |
| WCS_POSLIMIT_1 WCS_POSLIMIT_2 WCS_POSLIMIT_3 | Motion limits in the main axes in WCS. <br> Only practical in conjunction with the NC command \#GET WCS POSLIMIT [ 744] | Real | [mm, inch] | R |
| SIGNAL_READ | Read a signal without waiting [as of V2.11.2820.00] <br> 1: Signal present and read, | Boolean | 0,1 | R |


|  | 0: Signal not present <br> Only practical in conjunction with the NC command <br> \#SIGNAL READ [户 391] |  |  |
| :--- | :--- | :--- | :--- |
| RANDOM | Supplies a random value in the range 0.0 to 1.0 <br> Variable is not available under TwinCAT. | Real | - |

The following variables permit access to the data of kinematics defined in the channel parameter list.
In versions up to V 2.11 .28 xx and for single step transformations from V3.00, access uses V.G.KIN[j].* (where j:= kinematic ID).

For multi-step transformations (as of Build V3.00.3012.00), access uses V.G.KIN_STEP[i].ID[j].* (where $\mathrm{i}:=$ transformation step 0 or 1 and $\mathrm{j}:=$ kinematic ID).

## Attention

Write access causes a permanent change to internal channel parameter data. Value must be programmed in internal units.
The changes are retained until the next start-up or updating of the channel parameter list.

| (single-stage up to | Meaning | Data type | Unit for <br> input/out- <br> put | R/ <br> V3.00.3017.01): <br> V.G.KIN[j].* <br> (multi-stage up to <br> V3.00.3018.00): <br> V.G.KIN_STEP[i].ID[j].* |
| :--- | :--- | :--- | :--- | :--- |
|  |  |  |  |  |
| PARAM[k] | Kinematic parameter where <br> $<k>:=0 ~ \ldots . ~ M a x . ~ n u m b e r ~ o f ~ p a r a m e t e r s ~-1 ~$ | Real | $[0.1 \mu \mathrm{~m}$, R/W <br> $\left.10^{-4 \circ}\right]$  |  |

* Permitted access: R/W = Read/Write

In addition, the following variables listed are provided for access to the data of the Universal Kin-
ematic (ID 91, [FCT-C27])

| ZERO_ORIENTATION[k] | Initial orientation of the tool where <k>:= 0, 1, 2 (X, Y, Z) | Real | - | R/W |
| :---: | :---: | :---: | :---: | :---: |
| ZERO_POSITION[k] | Initial position of the tool where <k>:= 0, 1, 2 (X, Y, Z) | Real | $\begin{aligned} & {[0.1 \mu \mathrm{~m},} \\ & \left.10^{-4}{ }^{\circ}\right] \end{aligned}$ | R/W |
| PROGRAMMING_MODE | Programming mode, see P-CHAN-00112 | Integer | - | R/W |
| RTCP | Angle transformation | Boolean | - | R/W |
| NUMBER_OF_AXES | Number of axes in kinematic chain | Integer | - | R/W |
| AXIS[k].* | Axis of kinematic chain where <k>:= 0 ... V.G.NUMBER_OF_AXES-1 | - | - | - |
| AXIS[k].TYPE | Axis type (1: translatory, 2: rotary) | Integer | - | R/W |
| AXIS[k].ORIENTATION[l] | Orientation vector of axis (direction) where <l>:= 0, 1, $2(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ | Real | - | R/W |
| AXIS[k].POINT[I] | Interpolation vertex on axis where $\text { <\|>:= 0, 1, } 2(\mathrm{X}, \mathrm{Y}, \mathrm{Z})$ | Real | $\begin{aligned} & {[0.1 \mu \mathrm{~m},} \\ & \left.10^{-4 \circ}\right] \end{aligned}$ | R/W |
| CHAIN[k] | Description of axis sequence in the kinematic chain where <k>:= 0 ... V.G.NUMBER_OF_AXES-1 | Integer | - | R/W |

The 3 variables below permit read access to the currently selected kinematic ID:

| V.G.KIN_ID | Currently selected kinematic ID for single- <br> step transformations | Integer | - | R |
| :--- | :--- | :--- | :--- | :--- |
| V.G.KIN_ID_STEP[ <br> 0] | Currently selected kinematic of the first step for <br> multi-step transformations (as of Build <br> V3.00.3018.00) | Integer | - | R |
| V.G.KIN_ID_STEP[ <br> 1] | Currently selected kinematic of the second step <br> for multi-step transformations (as of Build <br> V3.00.3018.00) | Integer | - | R |
| TOTAL_KIN_OFF- <br> SET[i] | Effective total offset of the selected kinematics, <br> consisting of the sum of kinematic offsets of the <br> active tool and the kinematic offsets in the <br> channel parameter list. [as of V2.11.2024.00] | Real | $[0.1 \mu \mathrm{~m}$, <br> $\left.10^{-40}\right]$ | R |


| MAIN_FILE_NAME | File name of the NC main program. With MDI or manual block the variable supplies the filename "-". | String | - | R |
| :---: | :---: | :---: | :---: | :---: |
| MAIN_PROG_NAME | Name (\%...) of the NC main program | String | - | R |
| MAIN_PROG_NR | Number of the NC main program if the program name is a digit | Integer | - | R |
| FILE_NAME | File name of the currently active NC program | String | - | R |
| PROG_NAME | Name (\%...) of the currently active NC program | String | - | R |
| PROG_NR | Number of the currently active NC program if the NC program name is a digit | Integer | - | R |
| PROG_LEVEL | Program level of the currently active NC program: <br> 1: Main program <br> $\geq 2$ : Local or global subroutine | Integer | - | R |
| PATH_NR | Logical path number of the currently active NC main or global subroutine contained in the start-up list ( P -STUP-00019) | Integer | - | R |
| FILE_OFFSET | File position of start of the NC block where V.G.FILE_OFFSET is programmed | Integer | - | R |
| MIRROR_ACTIVE | Is mirroring (G351 X... or G21, G22, G23) active? If active, then 1 | Boolean | 0,1 | R |
| $\begin{aligned} & \text { BLOCK_SEARCH_AC } \\ & \text { TIVE } \end{aligned}$ | Block search active? If active, then 1 | Boolean | 0,1 | R |
| TRAFO_ACTIVE | Is kinematic transformation (\#TRAFO...) active? If active, then 1 | Boolean | 0,1 | R |
| MCS_ACTIVE | Is temporary transition to the machine axis coordinate system (\#MCS...) active? <br> If active, then 1 | Boolean | 0,1 | R |
| HSC_ACTIVE | Is free-form surface machining (\#HSC...) or spline interpolation (\#SPLINE... or G151) active? <br> If active, then 1 | Boolean | 0,1 | R |
| $\begin{aligned} & \text { HSC_SURFACE_ACT- } \\ & \text { IVE } \end{aligned}$ | Is free-form surface machining (\#HSC [SURFACE...]) active? If active, then 1 | Boolean | 0,1 | R |
| $\begin{aligned} & \text { CONT_MODE_ACT- } \\ & \text { IVE } \end{aligned}$ | Is polynomial contouring (G261) active? <br> If active, then variable supplies the value of the current contouring type: <br> 3 with contouring type DIST <br> 4 with contouring type DEV <br> 5 with contouring type POS <br> 6 with contouring type DIST_SOFT <br> 6 with contouring type DIST_MASTER <br> 7 with contouring type PTP <br> 0 with G260 | Integer | - | R |
| TRC_ACTIVE | Is tool radius compensation (G41, G42) active?? If active, then 1 | Boolean | 0, 1 | R |


| OTC_ACTIVE | Is online tool compensation (\#OTC...) active? If active, then 1 | Boolean | 0,1 | R |
| :---: | :---: | :---: | :---: | :---: |
| CAXTRACK_ACTIVE | C axis tracking (\#CAXTRACK...) active? If active, then 1 | Boolean | 0,1 | R |
| CYCLE_ACTIVE | Is the current program level a cycle? If the current program was called with L or LL CYCLE... or G8xx with parameter transfer, then 1 | Boolean | 0,1 | R |
| @P[i].VALID | Is transfer parameter programmed in the cycle call [> 210] or subroutine call with -G8xx? <br> 0 : Parameter is not programmed <br> 1: Parameter is programmed <br> The index contains the number of the parameter which is subject to a validity check. (E.g. 1 for @P1 or on request of the first parameter when called with G8xx). <br> NOTE for read access: The variable can only be used inside a cycle or in global subroutines G8xx with parameter transfer. | Boolean | 0,1 | R |
| TIME_STAMP | Current time stamp <date time> in format <DD.MM. YYYY hh:mm:ss:zzz>: <br> D: Day, two-digit <br> M: Month, two-digit <br> Y: Year, four-digit <br> h: Hours, two-digit <br> m: Minutes, two-digit <br> s: Seconds, two-digit <br> z: Milliseconds, three-digit <br> (e.g. 16.06.2015 14:08:10:123) [as of V2.10.1507.02] | String | - | R |
| TIME_STAMP_FILE_N AME | Current time stamp <date time> in the format as per ISO 6801:2004 <YYYYMMDDThhmmsszzz> without '.' and ':' e.g. for use in names of log files where: <br> Y: Year, always four-digit <br> M: Month, two-digit <br> D: Day, two-digit <br> h: Hours, two-digit <br> m : Minutes, two-digit <br> s : Seconds, two-digit <br> z: Milliseconds, three-digit <br> (e.g. 20150616T140810123) [as of V2.11.2024.03] | String | - | R |
| LIFT_ACTIVE | Is lift/lower an axis (X[LIFT...]) active? If active, then 1 | Boolean | 0,1 | R |

The following variables permit access to data of the $M / H$ functions defined in the channel parameters.

## Notice

When M/H functions are defined with pre-output function (MEP_SVS, MET_SVS), the following must be taken into consideration:

If write access $(S)$ to the pre-output values and the synchronisation modes of the $\mathrm{M} / \mathrm{H}$ functions do not match, the synchronisation mode is implicitly adapted.
If read access (L) to the pre-output values and the synchronisation modes of the $\mathrm{M} / \mathrm{H}$ functions do not match, an error message is output.

## Attention

The changes are retained until the next start-up or updating of the channel parameter list.

| M_FCT[i].SYNCH | Synchronisation mode of an M function <br> defined in the P-CHAN-00041 channel para- <br> meters < i> | Integer | - | R/W |
| :--- | :--- | :--- | :--- | :--- |
| M_FCT[i].PRE_OUTP_PATH | Pre-output path m_pre_output of an M func- <br> tion <i> of type MEP_SVS defined in the P- <br> CHAN-00070 channel parameters | Real | [mm, inch] | R/W |
| M_FCT[i].PRE_OUTP_TIME | Pre-output time m_pre_output of an M function <br> <i> of type MET_SVS in the P-CHAN-00070 <br> channel parameters | Real | [s] | R/W |
| H_FCT[i].SYNCH | Synchronisation mode of an H function <i> <br> defined in the P-CHAN-00027 channel para- <br> meters | Integer | - | R/W |
| H_FCT[i].PRE_OUTP_PATH | Pre-output path h_pre_output of an H function <br> <i> of type MEP_SVS defined in the P- <br> CHAN-00107 channel parameters | Real | [mm, inch] | R/W |
| H_FCT[i].PRE_OUTP_TIME | Pre-output time h_pre_output of an H function <br> <i> of type MET_SVS in the P-CHAN-00107 <br> channel parameters | Real | [s] | R/W |

The "V.G.SPEED_LIMIT" variables permit access to parameters pre-assigned in the channel parameters for speed limit look ahead [1] [ 819]-6.

## Notice

With write access the changed parameterisation is output to the channel. A possibly active motion is interrupted.
If write access $(S)$ to these values and the unit of the distance parameters do not match, the unit is implicitly adapted.
If read access $(\mathrm{L})$ to these values and the unit of the distance parameters do not match, an error message is output.

## Attention

The changes are retained up to program end or RESET. After program start the settings of the channel parameters are valid again.

| SPEED_LIMIT.ENABLE | Selecting/deselecting speed limit look ahead (0: deselect 1: select) | Boolean | 0,1 | R/W |
| :---: | :---: | :---: | :---: | :---: |
| SPEED_LIMIT.VEL_LIMIT | Definition of speed limit value | Integer | [\%] | R/W |
| SPEED_LIMIT.TIME | Defining the unit of the distances to and from the "corner" ( 0 : path 1: time) | Boolean | 0,1 | R/W |
| SPEED_LIMIT.DIST_TO_CORNER | Path distance to "corner" | Real | [mm, in ch] | R/W |
| SPEED_LIMIT.DIST_FROM_CORNER | Path distance from "corner" | Real | [mm, in ch] | R/W |
| SPEED_LIMIT.TIME_TO_CORNER | Time distance to "corner" | Real | [s] | R/W |
| SPEED_LIMIT.TIME_FROM_CORNER | Time distance from "corner" | Real | [s] | R/W |
| SPEED_LIMIT.OVERRIDE_WEIGHT | Weighting of speed limit value ( 0 : no weighting 1: weighting by override) | Boolean | 0,1 | R/W |


| MAX_NC_BLOCKS_AHEAD | Block-related decoder block ahead limit- <br> ing: all NC blocks | Integer | - | R/W |
| :--- | :--- | :--- | :--- | :--- |
| MAX_MOTION_BLOCKS_AHEAD | Block-related decoder block ahead limit- <br> ing: NC motion blocks only | Integer | - | R/W |
| MAX_TIME_AHEAD | Time-related decoder block ahead limit- <br> ing | Real | [s] | R/W |


| CIRCLE_CP_ABS | Circle centre point definition, 0: G162 active 1: G161 <br> active | Boolean | 0,1 | $R$ |
| :--- | :--- | :--- | :--- | :--- |
| CIRCLE_CPC_ACTIVE | Is circle centre point correction (G165) active? <br> If active, then 1 <br> 0 with G164 | Boolean | 0,1 | $R$ |


| SEGMENTATION_ACTIVE | Is segmentation (\#SEGMENTATION...) act- <br> ive? If active, then 1 | Boolean | 0,1 | R |
| :--- | :--- | :--- | :--- | :--- |
| STROKE_DEF_ACTIVE | Is definition of a stroke motion (\#STROKE <br> DEF...) active? If active, then 1 | Boolean | 0,1 | R |
| PUNCH_ACTIVE | Is online tool compensation (\#PUNCH...) act- <br> ive? <br> If active, then 1 | Boolean | 0,1 | R |
| NIBBLE_ACTIVE | Is nibbling (\#NIBBLE...) active? <br> If active, then 1 | Boolean | 0,1 | R |
| EDM_ACTIVE | Is EDM machining (\#EDM...) active? <br> If active, then 1 <br> [as of CNC Build V3.1.3019] | Boolean | 0,1 | R |
| FRICTION_ACTIVE | Recording friction compensation values <br> (\#FRICTION ON [...] active? <br> If active, then 1 <br> [as of CNC Build V2.11.2022.05] | Boolean | 0,1 | R |
| VECTOR_OFFSET_ACTIVE | Is cutting edge compensation (\#VECTOR <br> OFFSET...) active? If active, then 1 | Boolean | 0,1 | R |
| DIAMETER_PROG_ACTIVE | Is (G51) diameter programming active? <br> If active, then 1 <br> [as of CNC Build V2.11.2051.00] | Boolean | 0,1 | R |
| DIAMETER_PROG.ZERO_OFF- | Supplies the value of P-CHAN-00091 if the <br> following applies: <br> - Diameter programming (G51) is active <br> - Face turning axis is configured <br> [as of CNC Build V2.11.2051.00] | Boolean | 0,1 | R |
| SET |  |  |  |  |


| FILE_EXIST | Result of request with \#FILE EXIST [ 439] <br> If file check is positive, then 1 | Boolean | 0,1 | $R$ |
| :--- | :--- | :--- | :--- | :--- |


| MEAS_TYPE | Value of currently active measuring type [ 90] [as of <br> V2.11.2022.03] | Integer | - | $R$ |
| :--- | :--- | :--- | :--- | :--- |


| TRACK_CS.X | Values of master position with dynamic coordinate sys- | REAL | [mm, inch] | R |
| :--- | :--- | :--- | :--- | :--- |
| TRACK_CS.Y | tem (see [FCT-C30]). |  |  |  |
| TRACK_CS.Z |  |  |  |  |
| TRACK_CS.A |  |  |  |  |
| TRACK_CS.B |  |  |  |  |
| TRACK_CS.C |  |  |  |  |

The following V.G. variables permit the readout of the number of all write accesses to the contents of each list since controller start-up. [available as of Build V2.11.2037.03]

| CHAN_LIST_INVOKE | Channel parameter list | Integer | - | R |
| :--- | :--- | :--- | :--- | :--- |
| CLAMP_LIST_INVOKE | List of clamping offsets | Integer | - | R |
| TOOL_LIST_INVOKE | Tool list | Integer | - | R |
| VE_VAR_LIST_INVOKE | List of external variables | Integer | - | R |
| ZERO_POINT_LIST_ <br> INVOKE | Zero offset list | Integer | - | R |

The number of write accesses to individual zero offset groups is also possible [available as of V2.11.2037.03].

| NP[j].INVOKE_COUNT | Number of all write accesses to a zero offset since <br> controller start-up | Integer | - | $R$ |
| :--- | :--- | :--- | :--- | :--- |
| NP_AKT. <br> INVOKE_COUNT | Number of all write accesses to current zero offset <br> group since controller start-up | Integer | - | $R$ |

## Programing Example

Read the number of write accesses to a zero offset group

```
N010 P1 = V.G.NP[1].INVOKE_COUNT ; Read counter
N020 V.G.NP[1].V.X = 15
N030 V.G.NP[1].V.X = 16
N040 $IF P1 +2!= V.G.NP[1].INVOKE_COUNT ; Read counter again
N050 #ERROR [ID1]
NO60 $ENDIF
N070 G54
N080 $IF V.G.NP_AKT.INVOKE_COUNT != V.G.NP[1].INVOKE_COUNT
N090 #ERROR [IDD2]
```

kernel Industrielle Steuerungstechnik GmbH

N110 G53
N120 M30

The following V.G. variables for TCP velocity limiting are available as of V3.1.3079.26

| LIMIT.KIN[i].TOOL.LENGTH | Tool length where <br> $<i>:=0,1$ index of the configured kinematic | Real | $[\mathrm{mm}]$ | R/W |
| :--- | :--- | :--- | :--- | :--- |
| LIMIT.KIN[i].TOOL.KIN_PARAM <br> $[j]$ | Kinematic parameter of the tool where <br> $<j>:=0 . .69$ Index of the kinematic parameter | Real | $\left[0.1 \mu \mathrm{~m}, 10-4^{\circ}\right]$ | R/W |

### 14.3.1 Versioning of NC programs

NC programs can be provided with a version number using the V.G. variable V.G.PROG_VERSION.

## Notice

The complete version number must always have the format "<Major>.<Minor>.<Build>.<Patch>".

If a different format is used, the error ID 22015 is output.
The complete version number is composed of the following:
"Complete"="<Major>.<Minor>.<Build>.<Patch>"
For example, the complete version number or individual elements of the version number can be defined.
V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
V.G.PROG_VERSION.PATCH $=4$

| PROG_VERSION.MAJOR | Major version of the NC program <br> $($ e.g. 4 with 4.1.2.3 ) | UNS08 | - |
| :--- | :--- | :--- | :--- |
| PROG_VERSION.MINOR | Minor version of the NC program <br> $($ e.g. 1 with 4.1.2.3 ) | UNS08 | - |
| PROG_VERSION.BUILD | Build version of the NC program <br> $($ e.g. $\mathbf{2}$ with 4.1.2.3 ) | UNS08 | - |
| PROG_VERSION.PATCH | Patch version of the NC program <br> (e.g. 3 with 4.1.2.3 ) | UNS08 | - |
| PROG_VERSION.COMPLETE | Major.Minor.Build.Patch | RTRING | - |

The version information is inherited to called subroutines. A self-defined version number can also be assigned in subroutines and it is then inherited to its called subroutines.

## Example

## Versioning of NC programs

## Example 1

```
%L UP 1
N110 V.G.PROG_VERSION.COMPLETE = "5.1.2.3"
N120 #MSG ["Version UP_1: %s", V.G.PROG_VERSION.COMPLETE]
N130 M17
%MAIN
N010 V.G.PROG VERSION.COMPLETE = "4.1.2.3"
N20 LL UP 1
N30 #MSG ["Version Main: %s", V.G.PROG_VERSION.COMPLETE]
N040 M30
```

The following is output:
Version UP_1: 5.1.2.3
Version Main: 4.1.2.3

## Example 2

```
%L UP 1
( ---- no separate version specified ---)
N120 #MSG ["Version UP_1: %s", V.G.PROG_VERSION.COMPLETE]
N130 M17
%MAIN
N010 V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
N2O LL UP 1
N30 #MSG ["Version Main: %s", V.G.PROG_VERSION.COMPLETE]
NO40 M30
```

The following is output:
Version UP_1: 4.1.2.3
Version Main: 4.1.2.3

### 14.4 Self-defined variables (\#VAR, \#ENDVAR, \#DELETE)

Self-defined variables are created and possibly initialised in the NC main program or subroutine after the program name in a declaration block which starts with \#VAR and ends with \#ENDVAR.


## Programing Example

```
Create variables with and without type declaration
%test_var_def_1
:
#VAR
    V.P.VAR_1
    V.P.VAR 2 = 10.67
    V.P.VAR 3 : UNS32 = 10
    V.L.NAME 1 : STRING[20] = "BASEPLATE"
    V.L.VAR_1 : REAL = 23.45
    V.L.VAR_2 : SGN08
#ENDVAR
```

For a better overview, initialisation of a variable array can be written across sev-
eral NC lines using the "\" character.
\%test_var_def_2
:
\#VAR
V.P.ARRAY_1[3][6] $=[10,11,12,13,14,15, \$
$20,21,22,23,24,25, \$
$30,31,32,33,34,35$ ]
V.L.MY_ARRAY[3][6] $=[10,11,12,13,14,15,20,21,22,23,24,25,30,31,32,33,34,35]$
\#ENDVAR

## Notice

Access to array variables starts with index 0 . In the above example, access V.L.MY_ARRAY[0][5] then supplies the value 15 .

Self-defined variables and variable arrays can also be deleted in the NC program. The \#DELETE command is provided for this.

Syntax:
\#DELETE V.<name> \{, V.<name>\}

## Programing Example

In addition, the SIZEOF and EXIST functions are provided (see Section Arithmetical expressions <expr> [ 31]) to determine the dimensional size of variable arrays and check for the existence of self-defined variables.

## Programing Example

The EXIST request for a self-defined V.S. array variable (with any valid index) checks whether this variable was already defined in a previous NC program or if this variable must still be defined:

N10 \$IF EXIST[V.S.EXAMPLE[0]] == TRUE
N20 V.S.EXAMPLE[2] $=10$;assign a value to V.S. variable[2]
30 \$ELSE
N40 \#VAR
N50 V.S.EXAMPLE[5] $=[1,2,3,4,5]$
N60 \#ENDVAR
N70 \$ENDIF

### 14.4.1 Global, valid up to end of main program (V.P.)

The identifier "V.P. ...." permits the definition of self-defined variables which are detected (global) at the current program level and at all other program levels but are not valid after main program end. V.P. variables may be assigned values in REAL format. As of Build V2.11.2032.08 declarations of other data types [ $>$ 602] with initial values are possible.
V.P. variables can also be created as multi-dimensional arrays. A maximum of 4 dimensions is possible, e.g. V.P.TEST[1][2][3][4].
Syntax:
V.P.<FREE_DEF> global variable, not valid after (main) program end
<FREE_DEF> User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

## Programing Example

Global variable not valid after (main) program end (V.P.)

Create V.P. variables with and without assigning data types and initial values.

```
:
#VAR
```

    V.P.VAR_1 ;REAL64, 0.0
    V.P.VAR_2 : UNS32 = \(200 \quad\);UNS32, 200
    V.P.VAR_3 : REAL64 = 11.34 ;REAL64, 11.34
    V.P.VAR_4 : STRING[20] = "END_OK" ;STRING, END_OK
    V.P.VAR \({ }^{-}\)- \(=20\);REAL64, 20. \(\overline{0}\)
    \#ENDVAR
:
XV.P.VAR_5 ;X20.0

The maximum number of self-defined V.P. variables is fixed [6] [> 819]-6.21. At program start, all names and values of V.P. variables are deleted.


Variable validVariable not valid

### 14.4.2 Global, valid throughout main program (V.S.)

The code "V.S. ...." permits the definition of self-defined variables which retain the same names and the last values assigned at all program levels and all following (main) HC programs. After RESET, these variables also remain valid. The values of the variables can only be changed by overwriting. The variables themselves can only be deleted by \#DELETE or a controller restart. V.S. variables may be assigned values in REAL format. As of Build V2.11.2032.08 declarations of other data types [ ${ }^{\circ}$ 602] with initial values are possible.
V.S.<FREE_DEF> global variable, not valid after (main) program end
<FREE_DEF> User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

## Programing Example

## Global, valid (main) program global (V.S.)

Create V.S. variables with and without assigning data types and initial values.
: \#VAR
V.S.VAR1 ;REAL64, 0.0
V.S.VAR2 : UNS32 = 200 ;UNS32, 200
V.S.VAR3 : REAL64 = 11.34 ;REAL64, 11.34
V.S.VAR4 : BOOLEAN ;BOOLEAN, FALSE or 0
V.S.VAR [5] $=[5,10,10,15,20]$;Array with REAL64 values

## \#ENDVAR

:
XV.S.VAR[4]
; X20.0

The maximum number of self-defined V.S. variables is fixed [6] [> 819]-6.22.


Variable validVariable not valid

### 14.4.3 Local, valid throughout subroutine (V.L.)

The code "V.L. ...." permits the definition of self-defined variables which are valid local at the current program level and in directly called subroutines. They are deleted when the program level in which they were created is left (return).

A V.L. variable may be redefined with the same name at a lower program level and assigned a new value. This value is valid until this program level is left (return). The V.L. variable with the same name then has the original value.
V.L. variables may be assigned values in REAL format. As of Build V2.11.2032.08 declarations of other data types [ ${ }^{\circ}$ 602] with initial values are possible.
V.L. variables can also be created as multi-dimensional arrays. A maximum of 4 dimensions is possible, e.g. V.L.TEST[1][2][3][4].

Syntax:

```
V.L.<FREE_DEF>
local variable, not valid after (main) program end
```

<FREE_DEF> User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

## Programing Example

Local valid after subroutine end (V.L.)

Create V.L. variables with and without assigning data types and initial values. : \#VAR
V.L.LOC VAR1 ;REAL64, 0.0
V.L.LOC_VAR2 : UNS32 = 200 ;UNS32, 200
V.L.LOC_VAR3 : REAL64 = 11.34 ;REAL64, 11.34
V.L.LOC_VAR4 : BOOLEAN ;BOOLEAN, FALSE or 0
V.L.LOC_VAR5 = 10 ;REAL64, 10.0
\#ENDVAR
:
XV.L.LOC VAR5 ;X10.0

The maximum number of self-defined V.L. variables is fixed [6] [> 819]-6.23. At the start of the program, all names and values of those V.L. variables are deleted.


Variable validVariable not valid

### 14.4.4 Cycle variables (V.CYC.)

## Release Note

This function is available as of CNC Build V2.11.2032.08

## Notice

A precondition for using V.CYC. variables is that memory space must be reserved via the channel parameter P-CHAN-00418

The identifier "V.CYC. ..." addresses self-defined variables that must be used by preference within cycle programs. In addition, V.CYC. variables can also be used in standard main programs and subroutines. Besides specifying the variable name, the statement also includes a definition of the data type. The variables are valid as from their declaration in the current program level and in all other directly called program levels ( subroutines). They are deleted when they leave (return) the program level where they were created (see Validity and visibility [ 612]).
V.CYC. variables also have the option of creating multi-dimensional arrays. A maximum of 4 dimensions is possible, e.g. V.CYC.Test[1][2][3][4].

Syntax:
V.CYC.<FREE_DEF> cycle/program-specific variable
<FREE_DEF> User-defined name consisting of any number of characters (excluding blanks, tabulators, comments, comparison operators, mathematical operators, square brackets).

## Example

## V.CYC statement

```
%CYCLE TEST.cyc
P1 = 3 ;first index of the array
P2 = 2 ;second index of the array
P3 = 10 ;default maximum string length
#VAR
    V.CYC.TEST A[P1][P2] : STRING[P3]
    V.CYC.TEST B : STRING[P3] = "TEXT"
    V.CYC.TEST C : REAL64 = 1.0
#ENDVAR
:
M30
```


### 14.4.4.1 Validity and visibility

A V.CYC. variable is visible and usable starting with the declaration at the program level and in the program levels below it.


Fig. 151: Validity of self-defined V.CYC. variables

If a V.CYC: variable is created with the same name in a lower program level, the 'most local" declaration is visible and usable.


Fig. 152: Validity of V.CYC. variables of the same name

### 14.4.4.2 Delete variables

A V.CYC. variable is automatically deleted when the program exists the cycle or the program level in which the variable was created.

Alternatively, a V.CYC. variable can also be deleted by the NC command \#DELETE in the program level in which it was created.

For example,
\#DELETE V.CYC.MyVar

## Notice

\#DELETE can only delete V.CYC. variables in the program level in which it was created.

Deleting a non existing V.CYC. variable is ignored. No warning or error message is output.
If an attempt is made in a program level to delete an existing V.CYC. variable which was created in a higher program level, the error P-ERR-20933 is output.

### 14.5 External variables (V.E) (\#INIT V.E.)

The command "V.E. ...." writes to external addresses in the NC program and/or reads from external addresses. This is permitted by direct memory-linked communication between the NC channel and external users, typically the PLC.
Access from the NC channel can be executed synchronously by the interpolator or asynchronously by the decoder.

## Notice

i

## Reading a synchronous V.E. variable causes the NC channel to be flushed.

This is not permitted when TRC, polynomial contouring or HSC mode are active.

For more information on configuration and parameterisation, see the descriptions of external variables in the documentation [8] [> 819].

## Programing Example

```
N100 $IF V.E.EXT1 >= 100 (corr. to the value of V.E.EXT1)
N110 G01 X100 Y100 F1000
N120 $ELSE
N130 G01 X100 YV.E.EXT1 F1000 (linear interpolation in Y direction)
N140 ENDIF
N150 N150 = V.A.ABS.X (The external variable is assigned)
N160 G01 X
```

```
    (branched to)
```

    (branched to)
    (various cases)
    (various cases)
    (with the value of V.E.EXT1)
    (with the value of V.E.EXT1)
    (absolute X coordinates)
    (absolute X coordinates)
    ```
    (branched to)
```

    (branched to)
    (linear interpolation in X direction)
(linear interpolation in X direction)
(the value of V.E.EXT2)

```
    (the value of V.E.EXT2)
```

After controller start-up, the configured V.E. variables are initialised with zero.
Then V.E. variables can be re-initialised in the NC program using the \#INIT command. The command can be followed by one or several V.E. variables which are initialised completely. Besides individual V.E. variables, complete V.E. arrays, V.E. structures and subelements of V.E. structures can also be initialised.
\#INIT V.E.<name> \{, V.E.<name>\}

## Attention

## Access rights:

If a variable only has read access rights, it cannot be initialised with the \#INIT command. The same applies to V.E. structures if they contain at least one subelement that can only be read.

## Attention

## Synchronous V.E. variables:

As soon as a V.E. structure contains one synchronous variable, the entire initialization operation with \#INIT is synchronous, i.e. it is only executed in the interpolator context. Therefore, possible asynchronous subelements are also affected by this operation because they may not yet be reinitialised by a subsequent read access.
To achieve complete synchronism in these cases, users should therefore manually program a \#FLUSH WAIT command before the \#INIT command.
Tip:
When the \#INIT command is used, it is recommended to create V.E. structure variables so that all elements are completely synchronous or completely asynchronous.

## Programing Example

```
Initialisation of single V.E. variables:
Nxx #INIT V.E.EXT1, V.E.EXT2, V.E.EXT3
Initialisation of a V.E. array variable:
Nxx #INIT V.E.ARRAY1
```

Initialisation of specific V.E. array variables:
Nxx \#INIT V.E.ARRAY1[5], V.E.ARRAY1[8], V.E.ARRAY1[20]
Initialisation of a V.E. structure variable:
Nxx \#INIT V.E.STRUCT1
Initialisation of specific elements of a V.E. structure variable:
Nxx \#INIT V.E.STRUCT1.NBR POINTS, V.E.STRUCT1.POINTS
Combined initialisation of V.E. variables:
Nxx \#INIT V.E.EXT2, V.E.ARRAY1[5], V.E.STRUCT1.POINTS

## 15 Spindle programming

Spindle programming is performed in accordance with the standard syntax defined in ISO and ISO + extensions. This is required in particular for compatibility reasons and owing to certain standard functionalities such as turning, tapping and gear changing etc.

In order to comply with the requirements of new machine concepts and production technologies in flexible spindle programming, each spindle present on the channel also features the additional option of axis-specific programming.

This syntax allows several spindles to be addressed independently in multi-spindle systems in one NC block simultaneously (P-CHAN-00082, [6] [> 819]-8.8). In this case, it must be noted that only one spindle can be programmed at a time, the so-called "main spindle", both in the standard syntax and in the spindle-specific syntax.

All other spindles can be addressed only using the spindle-specific syntax (Section Spindle override in DIN syntax (G167) [> 668]).


Fig. 153: Correct use of DIN syntax and spindle-specific syntax

The spindles and the main spindle are defined in the channel parameter list [1] [ 819]-3. This configuration is available after controller start-up. The main spindle can be changed in the NC program by means of an NC command (\#MAIN SPINDLE, see Section Programmable spindle override [ 683]).

The table below shows what NC commands must be used only in the DIN syntax in conjunction with spindle programming and what NC commands are also permitted within the extended spindle-specific syntax.

## Overview of spindle commands

| Description | DIN syntax | Spindle-specific syntax |
| :--- | :--- | :--- |
| Spindle M functions | M3, M4, M5, M19 | M3, M4, M5, M19 |
| Speed | W | REV |
| Position | S.POS | POS |
| User-specific <br> M/H functions | Mxx/Hxx | Mxx/Hxx |
| Gear change (mechanical) | M40-M45 |  |
| Thread cutting | G33 |  |
| Tapping | G63 |  |
| Turning | G95, G96, G97, G196 |  |
| C axis | \#CAX | G112 |
| Gear change (new data) | G74 | G167 |
| Homing | G167 | CALLAX |
| Override 100\% |  | PUTAX |
| Explicit request for a spindle axis |  | GET_DYNAMIK_DATA |
| Explicit release of a spindle axis |  | FEED_LINK |
| Adopt tool dynamic data | G136, G137 |  |
| Feedforward control | Feed linking |  |

### 15.1 Parameterising spindles

### 15.1.1 Axis parameters

Axis channel parameter lists must be defined both for closed-loop controlled spindles and for open-loop controlled (PLC) spindles to enter the spindle-specific parameters [2] [ 819].

### 15.1.2 Channel parameter

Further entries which are made in the channel parameters are required for programming spindles in the NC program [1] [> 819].

In this case, each spindle to be addressed by this channel must be declared. For this purpose, a string (axis name) and the corresponding logical axis number are defined for each spindle. The axis names of the spindle can be freely selected but they must always start with "S" (e.g. S, S_MAIN, S1, SPINDEL_1).

In addition, the synchronisation modes must be defined spindle-specific for the spindle M functions (M3, M4, M5 and M19) and for the S word. The appropriate meaning of M3, M4, M5 and M19 must then be switched (P-CHAN-00098).

## Notice

The synchronisation method of the $S$ function has no effect if a spindle $M$ function is programmed in the NC block. Synchronisation only takes place based on the settings for the spindle $M$ function. The following priorities apply:
M19 > M3/M4/M5 > S

If the spindles are to be considered in the "Production time calculation" simulation mode, the data required for this can also be parameterised spindle-specific.

To ensure that this remains compatible with previous programming, one spindle must be declared as the main spindle ( $\mathrm{P}-\mathrm{CHAN}-00051, \mathrm{P}-\mathrm{CHAN}-00053$ ). The main spindle can then be programmed together with specific standard functionalities (e.g. tapping and gear changing etc.) in the conventional DIN syntax. Even if there is only one spindle in the system, it must be configured as the main spindle.

The optional gear changing function for the main spindle (P-CHAN-00052) is also enabled by setting a flag.

The default configuration defined in the channel parameters [1] [> 819] is provided after controller start-up.

## Example 1:

Configuration of a 1-channel system with 3 spindles. The spindle with axis number 6 is to be the main spindle. Gear change for this spindle is deactivated.

```
Channel parameter list [1] [ 819]:
```

```
:
```

:
spdl_anzahl 3
:
:
main_spindle_ax_nr
main_spindle_ax_nr
main_spindle_name
main_spindle_name
main_spindle_gear_change
main_spindle_gear_change

# 

# 

spindel[0].bezeichnung
spindel[0].bezeichnung
spindel[0].log_achs_nr
spindel[0].log_achs_nr
spindel[0].s_synch
spindel[0].s_synch
spindel[0].m\overline{3}_\mathrm{ synch}
spindel[0].m\overline{3}_\mathrm{ synch}
spindel[0].m4_synch
spindel[0].m4_synch
spindel[0].m5_synch
spindel[0].m5_synch
spindel[0].m19}_synch
spindel[0].m19}_synch
spindel[1].bezeichnung
spindel[1].bezeichnung
spindel[1].log_achs_nr
spindel[1].log_achs_nr
spindel[1].s_synch
spindel[1].s_synch
spindel[1].m3_synch
spindel[1].m3_synch
spindel[1].m4_synch
spindel[1].m4_synch
spindel[1].m5_synch
spindel[1].m5_synch
spindel[1].m1\overline{9}_synch
spindel[1].m1\overline{9}_synch
spindel[2].bezeichnung
spindel[2].bezeichnung
spindel[2].log_achs_nr
spindel[2].log_achs_nr
spindel[2].s_syynch
spindel[2].s_syynch
spindel[2].m\overline{3}_synch
spindel[2].m\overline{3}_synch
spindel[2].m4_synch
spindel[2].m4_synch
spindel[2].m5_synch
spindel[2].m5_synch
spindel[2].m1\overline{9}_synch
spindel[2].m1\overline{9}_synch
6 -> -> ->-
S |
S1 |
6-<-< -<-
spd__anzanl
spd__anzanl
3
0 |

```
:

After start-up, the spindle with the logical axis number 6 is the main spindle. It is addressed via the spindle name "S" and can be programmed in conventional DIN syntax or in spindle-specific syntax. Spindles "S2" and "S3" can only be programmed in spindle-specific syntax.


\section*{Example 2:}

Configuration of a 1-channel system with 3 spindles. The spindle with axis number 11 is to be the main spindle. Gear changing for this spindle is deactivated.
Channel parameter list [1] [ 819]:
:
```

spdl_anzahl

```
:
main_spindle_ax_nr
main_spindle_name
main_spindle_gear_change
\#
spindel[0].bezeichnung
spindel[0]. \(\log _{\text {_achs_nr }}\)
spindel[0].s_sȳnch
spindel[0].m3_synch
spindel[0].m4_synch
spindel[0].m5_synch
spindel[0].m19_synch
spindel[1].bezeichnung
spindel[1].log_achs_nr
spindel[1].s_sȳnch
spindel[1].m \(\overline{3}\) _synch
spindel[1].m4_synch
spindel[1].m5_synch
spindel[1].m19_synch
spindel[2].bezeichnung
spindel[2].log_achs_nr
spindel[2].s_sȳnch
spindel[2].m \(\overline{3}\) _synch
spindel[2].m4_synch
spindel[2].m5_synch
spindel[2].m1 \(\overline{9}\) _synch
3
11 -> -> -> ->
\(\mathrm{S}^{-}\)MAIN |
S1
:
6
\(0 \times 00000001\)
0x00000002
\(0 \times 00000004\)
\(0 \times 00000008\)
\(0 \times 00000001\)
S2 \(\quad-\quad-<-<\quad\) |
\(11-<-<-<-<\)
\(0 \times 00000001\)
0x00000002
0x00000004
\(0 \times 00000004\)
\(0 \times 00000008\)
\(0 \times 00000001\)
S3
30
x00000001
Configuration of a 1-channel system
main spindle. Gear changing for th
Channel parameter list [1] [ 819]:

\section*{Example 3:}

Configuration of a 2-channel system with a total of 3 spindles:
Channel 1: 3 spindles. Spindle with the axis number 11 is to be the main spindle.
Channel parameter list [1] [ 819]:
:
spdl_anzahl 3
:
main_spindle_ax_nr
11 -> -> -> ->
main spindle name
main_spindle_gear_change
\#
spindel[0].bezeichnung
spindel[0].log_achs nr
spindel[0].s_synch
spindel[0].m \(\overline{3}\) _synch
spindel[0].m4 synch
spindel[0].m5 synch
spindel[0].m19_synch
spindel[1].bezeichnung
spindel[1].log achs nr
spindel[1].s synch
spindel[1].m3_synch
spindel[1].m4 synch
spindel[1].m5-synch
spindel[1].m19_synch
spindel[2].bezeichnung
spindel[2].log_achs_nr
spindel[2].s synch
spindel[2].m3_synch
spindel[2].m4_synch
S
0

\section*{S1}

6
\(0 \times 00000001\)
0x00000002
0x00000004
\(0 \times 00000008\)
0x00000001
S2
11 -< -< -< -<
\(0 \times 00000001\)
0x00000002
0x00000004
\(0 \times 00000008\)
0x00000001
S3
spindel[2].m5-synch
30
spindel[2].m19_synch
:
\(0 \times 00000001\)
\(0 \times 00000002\)
0x00000004
\(0 \times 00000008\)
\(0 \times 00000001\)

Channel 2: 2 spindles. Spindle with the axis number 11 is to be the main spindle.
Channel parameter list [1] [ 819]:
:
```

spdl_anzahl 2
2

```
:
main_spindle_ax_nr
main_spindle_name
main_spindle_gear_change
\#
spindel[0].bezeichnung
spindel[0].log_achs_nr
spindel[0].s_synch
spindel[0].m \(\overline{3}\) _synch
spindel[0].m4_synch
spindel[0].m5_synch
spindel[0].m1 \(\overline{9}\) _synch
spindel [1].bezēichnung
spindel[1].log_achs_nr
spindel[1].bezeichnung
spindel[1].log_achs_nr
spindel[1].s_synch
spindel[1].m \(\overline{3}\) _synch
spindel[1].m4_synch
spindel[1].m5 synch
spindel[1].m1 \(\overline{9}\) synch
spindel[1].m5_synch
spindel[1].m19_synch
S1
6 -< -< -<-
\(0 \times 00000001\)
\(0 \times 00000002\)
\(0 \times 00000004\)
\(0 \times 00000008\)
\(0 \times 00000001\)
S2
11
\(0 \times 00000001\)
\(0 \times 00000002\)
11 -> -> ->-
S |
0
:
0
-
\(6-<-<-<-\)
\(0 \times 00000004\)
\(0 \times 00000008\)
:

After start-up, the spindle with the logical axis number 11 can be addressed as the main spindle by spindle name "S" for both channels. It can be programmed in conventional DIN syntax or in spindle-specific syntax. Spindle "S1" can also be programmed from both channels in spindle-specific syntax. Spindle "S3" is only available in channel 1 . This spindle is not known in channel 2.


\subsection*{15.2 Programming in DIN syntax}

\subsection*{15.2.1 \(\quad\) The spindle \(M\) functions}

\subsection*{15.2.1.1 Move spindle in DIN syntax ((M3/M4/M5)}
\begin{tabular}{|llc|}
\hline M03 & Spindle rotation clockwise (cw) & (modal) \\
M04 & Spindle rotation counter-clockwise (ccw) & (modal) \\
M05 & Stop spindle & (modal) \\
\hline
\end{tabular}

The spindle M functions M03, M04 and M05 define the spindle operation mode and must be used in conjunction with the S word (Sec. Spindle speed (S word) [ \(\downarrow\) 627]. They are modal and each may only be programmed alone in the NC block.
Spindle rotation is activated if M03 or M04 is programmed and a valid speed is set.
M05 stops spindle rotation. Note that this spindle M function is the default spindle mode after controller start-up and initial program start.
If no M05 is set at program end, the spindle continues to rotate.

\section*{Programing Example}

\section*{Move spindle (M3, M4, M5)}
```

N10 S1000 (Speed 1000 rpm is stored, none) (Spindle
rotation because M05 is default)
N20 M03 (Spindle rotation cw at 1000 rpm)
N30 M04 (Spindle rotation ccw at 1000 rpm)
N40 S500 (Spindle rotation ccw at 500 rpm)
N50 M05 S300 (Spindle stop, speed 300 rpm is)
N60 M04 (Spindle rotation ccw at 300 rpm)
N70 M05 (Spindle stop)
N80 M03 S1000 (Spindle rotation cw at 1000 rpm)
N90 M30 (Program end)

```

Channel parameter list [1] [> 819]:
The synchronisation modes must be defined spindle-specific for M3, M4, M5. The M function is not executed for synchronisation mode "0" (NO_SYNCH).
:
spindel[0].bezeichnung

S1
spindel[0].bezeichnung
spindel[0].log_achs_nr
spindel[0].s_synch
spindel[0].m3_synch
spindel[0].m4_synch
spindel[0].m5_synch
spindel[0].m1 \(\overline{9}\) _synch

6
\(0 \times 00000001\)
\(0 \times 00000002\)
0x00000002
\(0 \times 00000008\)
\(0 \times 00000001\)

\subsection*{15.2.1.2 Position spindle in DIN syntax ((M19,*. POS)}
\begin{tabular}{|llc|}
\hline M19 & Position spindle & (non-modal) \\
*.POS & Spindle position & (modal) \\
\hline
\end{tabular}

Spindle positioning can be represented by the following syntax:
```

M19 [ <Spindelname>.POS=.. ] [ M03 | M04 ] [ <SpindeIname>=.. ]

```

M19 Positioning spindle
<Spindelname>.POS=<Position>
<Spindelname>=<Drehzahl> CHAN-00053

Spindle position in [ \({ }^{\circ}\) ]. Designation of the main spindle as per P -
Spindle speed in [rpm].. Designation of the main spindle as per P-CHAN-00053

M03/M04 or spindle speed in the same NC block are optional. However, a valid spindle speed (> \(0)\) must be set.

The spindle position is modal and need not be respecified if M19 is reprogrammed. If no spindle position was previously programmed, the motion is moved to position "zero" by default.
If the spindle is not rotating, positioning is executed with the shortest motion path.
Spindle positioning with M19 is only permitted for position-controlled spindles (closed loop).

Programing Example
Positioning spindle (M19, *.POS)
The spindle is positioned to \(180^{\circ}\) in each case in the following examples. The "=" character is optional:
```

M19 S.POS180
M19 S.POS 180
M19 S.POS=180
M19 SPINDEL.POS=180
M19 S1.POS=180

```

The spindle does not rotate during positioning. The shortest motion path is calculated.
```

(Spindle stop, speed 100 rpm is stored)
(is stored as default)
(Position at 100 rpm at position 180)
(The direction of rotation results from the)
(The direction of rotation results from the)
(shortest motion path)
(Position at 200 rpm ccw at position 90)

```

Channel parameter list [1] [> 819]:
The synchronisation mode must be defined spindle-specific for M 19 . The M function is not executed for synchronisation mode "0" (NO_SYNCH).
:
\begin{tabular}{ll} 
spindel[0].bezeichnung & S 1 \\
spindel[0].log_achs_nr & 6 \\
spindel[0].s_synch & \(0 \times 00000001\) \\
spindel[0].m3_synch & \(0 \times 00000002\) \\
spindel[0].m4_synch & \(0 \times 00000002\) \\
spindel[0].m5_synch & \(0 \times 00000008\) \\
spindel[0].m19_synch & \(0 \times 00000001\)
\end{tabular}

\subsection*{15.2.2 Spindle speed (S)}
<Spindelname> [ = ] <Drehzah/>
(modal)
\begin{tabular}{ll} 
<Spindelname> & Designation of the main spindle as per P-CHAN-00053 \\
<speed> & Spindle speed \(1000[\mathrm{rpm}]\).
\end{tabular}

In the configuration, the main spindle can be assigned a string in the channel parameter list ( P -CHAN-00053). In order to avoid ambiguities, there must be an equals sign in front of the speed entry after all spindle names consisting of more than one character.

Values can be assigned to the \(S\) word directly or by means of parameters. Decimal numbers are also permitted (REAL format).
A distinction is made between the following types of use of the \(S\) word in conjunction with spindle M functions:
1. S word in conjunction with M03, M04, M19:

If the S word or the string used for it is programmed in conjunction with M03/M04 or M19, the value following the \(S\) word is interpreted as the spindle speed and is output to the spindle.
2. S word in conjunction with M05:

The value following the \(S\) word is adopted as the spindle speed in the working data in conjunction with M05, but it is not output to the spindle.

The \(S\) word alone does not generate a motion in the NC program. This requires that a spindle mode M03, M04, M19 is known. Accordingly, programming M03, M04 and M19 only causes motion when the S word is set (> 0, analogous to G01, G02 and G03 in which traversing only occurs when the feedrate and the axes to be traversed are specified).

\section*{Notice}

In the case of a negative \(S\) value, an error message is output.
A negative \(S\) value is only permitted in conjunction with G63 (tapping) since it triggers reversal of the direction of rotation on withdrawal out of the thread bore.

\section*{Programing Example}

\section*{Programming with spindle S}
```

N10 S300 (Speed 300 rpm is stored)
N20 M04 (Spindle rotation ccw at 300 rpm)
N30 M03 S1000 (Spindle rotation cw at 1000 rpm)
N40 S500 (M03 active, spindle rotation cw at 500 rpm)
N50 M05 S100 (Spindle stop, speed 100 rpm is stored)
N60 M04
N70 M05 (Spindle stop)
N80 M30 (Program end)

```

Channel parameter list [1] [> 819]:
The synchronisation mode must be defined spindle-specific for the \(S\) word. An error message is generated in the case of synchronisation mode "0" (NO_SYNCH) since an S word may not be ignored.
:
spindel[0].bezeichnung
\begin{tabular}{ll} 
spindel[0].log_achs_nr & 6 \\
spindel[0].s_synch & \(0 \times 00000001\) \\
spindel[0].m3_synch & \(0 \times 00000002\) \\
spindel[0].m4_synch & \(0 \times 00000002\) \\
spindel[0].m5_synch & \(0 \times 00000008\) \\
spindel[0].m19_synch & \(0 \times 00000001\)
\end{tabular}

\section*{Programing Example}

\section*{Spindle speed (S word)}
\begin{tabular}{|c|c|c|c|c|}
\hline N10 & M03 & \multicolumn{2}{|l|}{\multirow[t]{2}{*}{S100
S.P0S90}} & (Spindle rotation at 100 rpm) \\
\hline N20 & M19 & & & (At 100 rpm cw to position 90) \\
\hline N30 & M0 4 & \multicolumn{3}{|r|}{(Spindle rotation ccw at 100 rpm)} \\
\hline N40 & M19 & S200 & \multicolumn{2}{|l|}{S.POS 180 (At \(200 \mathrm{rpm} \mathrm{cw} \mathrm{to} \mathrm{position} \mathrm{180)}\)} \\
\hline N50 & M05 & \multicolumn{3}{|l|}{S150 (Spindle stop, speed 150 rpm)} \\
\hline N60 & M19 & \multicolumn{2}{|l|}{S.POS \(=135\)} & (Position at 150 rpm on) \\
\hline N70 & M03 & S300 & & (Spindle rotation cw at 300 rpm ) \\
\hline N80 & M19 & S200 & S.POS270 & (At 200 rpm cw to position 270) \\
\hline N90 & M03 & S400 & S.POS 45 & (Spindle rotation cw at 400 rpm ) (position 45 is stored) \\
\hline N100 & M19 & & & (At 400 rpm cw to position 45) \\
\hline N110 & M0 4 & S800 & & (Spindle rotation ccw at 800 rpm) \\
\hline N120 & S1200 & & & (Spindle rotation ccw at 1200 rpm) \\
\hline N130 & M5 & & & (Spindle stop) \\
\hline N140 & M03 & & & (Spindle rotation cw at 1200 rpm) \\
\hline N150 & M19 & & & (At 1200 rpm cw to position 45) \\
\hline
\end{tabular}

\subsection*{15.2.3 Spindle gear change (M40-M45)}

A spindle gear change is programmed with M40...M45. These M functions define max. six gear stages. A separate gear data record must be parameterised for each gear stage in the associated spindle axis list [AXIS].

The \(M\) functions can be programmed together with the spindle speed and the \(M\) function for the direction of rotation in the same NC block. Functions M40 to M45 are used to select the gear stage and to trigger mechanical gear change.
```

M40 | M41 | M42 | M43 | M44 | M45 [<spindle name>=.. ] [ M03 | M04 ] (modal)

```
M40 to M45 Gear stages 1 to 6
<Spindle name>=<expr>

Spindle speed consisting of spindle name according to P-CHAN-00053 and speed value in [rpm].

\section*{Programing Example}

Spindle gear change (M40-M45)

S800 M41 M03 (Speed 800 rpm, gear stage 2, rotation cw)
- The decoded functions M40 to M45 are modal and are activated at block start. M40 to M45 cancel each other out.
- The system permits definition of max. 6 spindle gear stages (M40... M45). The minimum and maximum speeds are defined for each gear stage in a "Table of speed stages" in [1] [ 819]-4 (unit = rpm).
- The maximum speed for the 10 V output in the case of closed-loop position-controlled spindles is defined for analogue drives via the Multi-Gain Factor P-AXIS-00128 and P-CHAN-00129..
- In a system with automatic gear stage selection, this is determined solely by programming the speed S. M40 to M45 then do not need to be programmed.
- M40 to M45 can only be programmed for closed-loop position-controlled spindles.
- The NC kernel always attempts to minimise gear changing (for example, if a new speed can be used with the current gear stage, a gear change is suppressed even if it was explicitly programmed with M40 to M45).

Channel parameter list [1] [ 819]:
- Definition of M functions M40-M45 and definition of synchronisation modes. :
\begin{tabular}{|c|c|c|}
\hline m_synch [1] & \(0 \times 00000001\) & MOS \\
\hline m_synch [2] & 0x00000002 & MVS \\
\hline m_synch [40] & \(0 \times 00000002\) & MVS \\
\hline m_synch [41] & \(0 \times 00000002\) & MVS \\
\hline m_synch [42] & 0x00000002 & MVS \\
\hline m_synch [43] & 0x00000002 & MVS \\
\hline m_synch [44] & \(0 \times 00000002\) & MVS \\
\hline m_synch [45] & 0x00000002 & MVS \\
\hline m_synch [48] & \(0 \times 00000008\) & MNS \\
\hline m_synch [49] & \(0 \times 00000002\) & MVS \\
\hline
\end{tabular}
- Activate gear change:
main_spindle_gear_change 1 0:OFF \(1:\) ON
- Parameterise the spindle gear (search direction, speed ranges):
:
spindel[0].range_way 0 :bottom up \(1:\) top down
\#
spindel[0].range_table[0].min_speed
spindel[0].range_table[0].max_speed
(M40)
spindel[0].range_table[1].min_speed
\(560 \quad\) (M40)
spindel[0].range_table[1].max_speed
400
(M40)
(M41)
spindel[0].range_table[2].min_speed
700 (M42)
spindel[0].range_table[2].max_speed
3500 (M42)
spindel[0].range_table[3].min_speed
3501
4000
spindel[0].range_table[3].max_speed
3800 (M44)
spindel[0].range_table[4].min_speed
\(5500 \quad\) (M44)
spindel[0].range_table[4].max_speed 5400 spindel[0].range_table[5].min_speed (M45) spindel[0].range_table[5].max_speed 7000 (M45) \#
:

\section*{Programing Example}

\section*{Spindle gear change (M40-M45)}

\section*{Automatic determination of the gear stage: ON}
:
spindel[0].autom_range 1
:

NC program:
S650 M03 OK, M41 \(\square\) PLC
S750 OK, no change, M41 already selected
S950 OK, automatic change, M42=> PLC
S1050 OK, no change, M42 already selected
S 750
OK, automatic change, M41=> PLC
OK, no change, M41 already selected

Error, speed too high

A programmed gear stage is always checked:
M41 S750 OK, "automatic" change, M41=> PLC
. . .but
M40 S750 Error, incorrect gear stage
\(\qquad\)

\section*{Programing Example}

\section*{Spindle gear change (M40-M45)}

Automatic determination of the gear stage: OFF
```

:
spindel[0].autom_range 0
:
NC program:
M41 S650 M03 OK, M41 => PLC
M41 S750 OK, no change, M41 already selected
M42 S950 OK, change, M42=> PLC
M42 S1050 OK, no change, M41 already selected
M41 S750 OK, change, M41=> PLC
M41 S500 OK, no change, M41 already selected
M41 S200 Error, program different gear stage (M40)
S950 Error, no gear stage (M42) programmed

```

\subsection*{15.2.4 Turning functions}

\subsection*{15.2.4.1 Diameter programming (G51/G52)}
\begin{tabular}{|lll|}
\hline G51 & Select diameter value & (modal) \\
G52 & Deselect diameter value & (modal, initial state) \\
\hline
\end{tabular}


Fig. 154: Reference points and diameter programming for turning
\begin{tabular}{ll} 
W Workpiece zero point & P Cutting point \\
X Face turning axis & Z Longitudinal turning axis \\
M Machine zero point & A Fixed stop point \\
C Control zero point & d Programmed dimension for diameter \\
& programming
\end{tabular}

When diameter programming is selected, the positional values for the face turning axis in a motion block are interpreted as diameter values relative to the turning centre point.

It should be noted that the programmed coordinates of the face turning axis only correspond to the workpiece diameter if the zero point of the face turning axis is located at the turning centre point (irrespective of whether offsets act as a diameter; see Programming example).

The axis parameters can parameterise axes in "face turning" mode:
- G51 with absolute programming (G90) (P-AXIS-00058)
- ... and/or G51 with relative programming (G91) (P-AXIS-00059
- Reference point programming (G92) and zero point programming (G53-G59) in the diameter (P-CHAN-00091)
G51 acts on the axis which is operated in "face turning" mode. When the face turning axis is selected, the face turning axis must exist in the machining plane (G17, G18, G19).

The coordinates of the circle centre point \((I, J, K)\) and circle radius programming \((R)\) are not programmed in the diameter.
Diameter programming is deselected with G52.

\section*{Programing Example}

Diameter programming (G51/G52) in G18
```

;General settings (optional):
;Display position values in the diameter P-CHAN-00256 (TRUE, 1)
;Settings of X axis:
;Face turning axis, translatory: P-AXIS-00015 (0x41)
;G51 with G90: P-AXIS-00058 (TRUE, 1)
;G51 with G91: P-AXIS-00059 (FALSE, 0) (optional)
;G92, G53-G59 in the diameter: P-CHAN-00091 (TRUE, 1) (optional)
;Settings of Z axis:
;Longitudinal turning axis, translatory: P-AXIS-00015 (0x81)
:
N05 G18
N10 G90 G01 F1000
N20 G51 X80 ;Diameter 80 mm
N30 G92 X10 ;G92 by 10 mm in the diameter
N40 X0 ;Position 0 + G92 => diameter 10 mm
N50 G91 X50 ;X relative +50 mm, not in the diameter
N80 G52 ; Deselect diameter programming

```

\subsection*{15.2.4.2 Cutter radius compensation (G40/G41/G42)}
\begin{tabular}{|llr|}
\hline G40 & Deselect SRK & (modal, initial state) \\
G41 & SRK left of contour & (modal) \\
G42 & SRK right of contour & (modal) \\
\hline
\end{tabular}

Tool tip radius compensation (SRK) acts in the machining plane selected using G17, G18, G19 for turning work. In this plane, one of the axes must be operated in "face turning" mode and the other in "longitudinal turning" mode. ( axis mode: P-AXIS-00015)
The data records stored in the D words are used as tool compensation values. For turning tools, the orientation of the cutter edge relative to the machining plane (face/longitudinal turning axis) must be specified in the parameter P-TOOL-00002 by an additional identifier 1... 9 (see figure).


Fig. 155: Orientation of cutter edge to machining plane.

A typical turning tool is characterised by the following values/parameters:
- Tool type
- SRK orientation
- Tool radius
- Tool length
- Tool offset (see figure below)

1 (turning tool)
1... 9

Tool tip radius
--


Fig. 156: Tool gauging for tool offset compensation.

When specifying tool axis offsets, their mathematical sign should be noted since it refers to components of the tool offset vector in the machining plane. In the example of a turning tool shown in the above figure, the offsets in the direction of the \(X\) and \(Y\) axis both have negative (mathematical) signs.

Tool offsets must be specified up to the theoretical tool top (point P).
A change between turning tool and milling tool is permitted when G41 or G42 is selected. With absolute programming (G90) the current axis offset values of the new tool are included in the calculation for the next motion block depending on the tool type.

\subsection*{15.2.4.3 Feedrate per revolution (G95)}
G95 Feedrate in \(\mathrm{mm} /\) revolution (modal)

During turning with active G95, a constant chip thickness can be defined with the F Word in [mm/ rev, inch/rev] irrespective of spindle speed (rpm).

Here, the axis feedrate is linked to the rotational speed (rpm) of the position-controlled spindle. It is only valid in conjunction with the \(G\) function with which it was programmed. This means that if a change is made from G95 to G94 or G93 (Section "Machining time or feedrate" [ 157]) the valid F word for G95 is not adopted..

G95 can also be programmed in combination with a controlled (PLC) main spindle ( P -CHAN-00069). Please note here that the resulting feedrate per revolution is only dependent on the programmed values (F word, S word). Real-time influences on the spindle speed are not considered, e.g. override changes. This type of programming in only permitted in a channel in conjunction with the assigned main spindle as of Build V3.1.3066.02.

\section*{Programing Example}

Feedrate per revolution (G95)
```

N10 F1000 G01 X0 Z100 M3 S1200 ; Feedrate 1000 mm/min (G94)
N20 G95 F1.5 ; Feedrate per revolution 1.5 mm/rev
; speed 1200 rpm
N30 Z50 ; Feedrate 1800 mm/min
N40 G94 X50 ; Feedrate }1000\mathrm{ mm/min valid from N10
N50 G93 F20 X20 ;Machining time 20 s
N60 G95 Y200 S2000 ; Feedrate per revolution 1.5 mm/rev valid from N20,
; speed 2000 rpm
N70 M30

```

\section*{Feedrate per revolution (G95) and axis coupling with variable gear ratio (gear coupling)}

As of Build 3.1.3079.03 the feedrate of the path axes can be coupled to the speed of the main spindle with \#TURN [ROT_FEED_CPL=1]. The feedrate is adapted depending on the gear ratio settings of the axis couplings. This permits the activation of the feedrate per revolution (G95) while spindles are rotating.

Axis couplings with variable gear ratios can be linked by
- the NC command \#GEAR LINK [ 486] or
- the HLI [FCT-A9// Axis coupling via HLI]

In the \#TURN [ROT_FEED_CPL=0] setting, the speed of the main spindle is used without any consideration given to axis couplings.
If the main spindle is changed, the setting of ROT_FEED_CPL is adopted automatically for the new main spindle and deactivated for the previous main spindle.

\section*{Programing Example}

\section*{G95 - Spindle speed with gear coupling}
```

;Axis 5 is configured as main spindle, Axis 6 (S2) is configured as auxiliary
spindle.
; ---------------------------------------------------------
; Axis 5 is coupled to Axis 6 and to itself.
N10 \#GEAR LINK ON [TARGETNR=5 AXNR1=6 AXNR2=5 NUM1=1 DENOM1=1 NUM2=1 DENOM2=1]
N20 M3 S500 ; commanded speed of Axis 5 = 500 rpm
N30 N30 [M3 REV1500] ; commanded speed of Axis 6 = 1500 rpm
; speed of Axis 5 = 500 rpm + 1500 rpm = 2000 rpm
; speed of Axis 6=1500 rpm
; feedrate = 1000 mm/min
N40 F1000 G01 X100
; Spindle speed related to axis coupling is used for G95
N50 \#TURN [ROT_FEED_CPL=1]
N60 G95 F1.5 ; feedrate per revolution = 1.5 mm/rev
; feedrate = 2000 rpm * 1.5mm/rev = 3000 mm/min
N70 X200
; Spindle speed without axis coupling is used for G95
N80 \#TURN [ROT FEED CPL=0]
; feedrate = 5000 rpm}* 1.5mm/rev = 750 mm/mi
N90 X400
N100 M05
N110 SC[M05]
N120 \#GEAR LINK OFF [TARGETNR=5]
M30

```

\subsection*{15.2.4.4 Constant cutting speed (G96/G97/G196)}

G96 Selecting constant cutting speed (modal)
G97 Deselecting constant cutting speed, selecting spindle speed (modal, initial state)
G196 Maximum spindle speed for G96 (G196 non modal, max. speed modal)

Using the G functions G96, G97 and G196, it is possible to optionally change the interpretation of the \(S\) word (or \(S\) strings):

G96 S in [m/min or ft/min *] (cutting speed)
G97 S in [rpm] (spindle speed)
G196 S in [rpm] (max. spindle speed during G96)
* [as of Build V2.11.2032.08 with G70 and P-CHAN-00360 = 1]

When selected with G96, the starting rpm of the spindle is calculated from the programmed cutting speed and the distance of the tool tip to the turning centre point. This distance results from the last (not in the current NC block) programmed position and the reference point offset of the face turning axis. Exactly one face turning axis must be present in the current machining plane (G17, G18, G19).

A cutting speed programmed for G96 using the \(S\) word is only valid until it is deselected by G97. With G96, constant cutting speed is only activated when the S word is programmed.

Specifying a maximum spindle speed with G196 in conjunction with the \(S\) word is optional and only active during G96. Spindle speed limiting must be programmed before G96 is selected.


Fig. 157: Spindle speed with active G96

\section*{Release Note}

Extended G function G196
As of Build V3.1.3057.04

Alternatively, the maximum spindle speed can be programmed as an additional value in [rpm] in conjunction with G196. It is modal.
This syntax permits the programming of G196 and G96 in the same NC block. A separate specific NC block is not required.

G196 = <max_spindle_speed>
(G196 non-modal, max. speed modal)

Close to the turning centre point, the programmed maximum spindle speed (G196) or the maximum spindle speed specified in the assigned axis parameters P-AXIS-00212 defines the limits of the constant cutting speed.
When deselected with G97, the last spindle speed set is retained.
Motion blocks of the face turning axis in rapid traverse (G00) lead to an interruption of G96 to prevent undesired speed value changes when the tool is positioned. The next motion block with G01, G02 or G03 cancels suppression of G96.

\section*{Programing Example}

Constant cutting speed (G96/G97/G196)
```

; X is the face turning axis
N10 M03 S1000 G01 F1500 X100
N20 G196 S6000 ;max. speed 6000 rpm
N30 G96 S63 ; select const. cutting speed 63 m/min,
;workpiece radius 100 mm corresp. to X coordinates
N40 X80
N50 S4 X50 ; new cutting speed 4 m/min; workpiece radius 80 mm,
;at block end 50 mm
N60 G97 ;max. speed 6000 rpm not effective here!
N80 G92 X-10 ;reference point offset in X by -10 mm
N90 G96 X60 ; cutting speed from N50 not valid: const.
;cutting speed not active, speed 8000 rpm
N100 S25 X70 ;cutting speed 25 m/min, workpiece radius 50 mm,
; (=60mm+BPV), const. cutting speed active
N110 G00 X450 ;rapid traverse: speed remains constant
N115 X70
N120 G01 X40 ; suppress G96 cancelled
N110 M30

```

\subsection*{15.2.4.5 Thread cutting with endlessly rotating spindle (G33)}

\section*{Single-start/multi-start threads}

When thread cutting with an endlessly rotating spindle (G33), the path motion is synchronised to the zero passage of the spindle rotation. Therefore, the thread can also be cut in several passes in succession. When an offset angle is specified as option, multi-start threads can also be produced.

To achieve a good machining result and to minimise path errors, feedforward control can be selected for the spindle and for path axes.

\section*{Programming}

\section*{Syntax example for ZX plane (longitudinal axis Z, feed axis X):}

G33 Z<expr> K<expr> [ <spindle_name>OFFSET[ = ]<expr>] (modal)

G33 Thread cutting with endlessly rotating spindle. The G33 function is modal. The next motion block with a modal block type (G00, G01, G02, G03, spline, polynomial) deselects thread cutting.
Z<expr> Target point ("thread length") in [mm, inch]
K<expr> The thread pitch is programmed with active thread cutting in the unit [mm/rev, inch/rev] without a mathematical sign using the address letters \(I\), \(J\) and \(K\). They are assigned to the \(X, Y\) and \(Z\) axes according to DIN 66025.
The thread pitch is modal up to program end and should not be zero on when \(G 33\) is selected. The feed is not programmed using the F word but results from the spindle speed and the thread pitch.
The pitch of longitudinal or tapered threads at an inclination angle less than \(45^{\circ}\) is specified by the address letter K if the Z axis is the longitudinal turning axis. With facing or tapered threads with an inclination angle greater than or equal to \(45^{\circ}\), the pitch is specified by I if the X axis is used as the face turning axis, and by J if the Y axis is used. The figure below shows examples for specifying thread pitch using the address letters in the Z-X plane.
<spindle_name>.OFFSET=<expr>
Thread offset angle in \(\left[{ }^{\circ}\right]\) in spindle modulo range. Only required as an option for multi-turn threads. The offset angle is modal up to program end. Spindle name according to P-CHAN-00053. The "=" character is optional.

Pitch values I, K with longitudinal thread


Fig. 158: Value of thread pitch for longitudinal thread

Pitch values I, K with tapered thread


Fig. 159: Value of thread pitch for tapered thread

\section*{Programing Example}

\section*{Thread cutting with endlessly rotating spindle (G33)}

G33 Z.. K.. [S.OFFSET..]


Fig. 160: Illustration of geometry example
Cutting a longitudinal thread (M70x1.5) with several cuts:
```

%L longit_thread

| N100 G33 Z48 K1.5 | ; Cut thread turn |
| :--- | :--- | :--- |
| N110 G00 X72 | ; Retract and move |
| N120 Z105 | ; to start position |
| N130 M29 | ; Subroutine end |

%G33 (thread depth 0.92 mm)
N10 G51
N15 T1 D1 M03 S400 ; Select tool, start spindle
N20 G00 X72 Z105 ;Start
N25 G01 X69.54 F1000 ;Position at 1st cutting depth
N30 LL longitudinal thread
N35 G01 X69.08 ;Position at 2nd cutting depth
N30 LL longitudinal thread
N35 G01 X68.62
N30 LL longitudinal thread
N35 G01 X68.16
N30 LL longitudinal thread
N35 G01 X68.16
N30 LL longitudinal thread

```
```

;Select diameter programming

```
;Select diameter programming
```

;Cut thread turn

```
;Cut thread turn
to start position
to start position
;Subroutine end
;Subroutine end
    ;1st thread cutting
    ;1st thread cutting
    ;1st thread cutting
    ;2nd thread cut
    ;2nd thread cut
    ;2nd thread cut
;Position at 3rd cutting depth
;Position at 3rd cutting depth
;Position at 3rd cutting depth
    ;3rd thread cut
    ;3rd thread cut
    ;3rd thread cut
;Position at final depth
;Position at final depth
;Position at final depth
    ;4th thread cut
    ;4th thread cut
    ;4th thread cut
;Reposition at final depth
;Reposition at final depth
;Reposition at final depth
    ;Empty cut
```

    ;Empty cut
    ```
    ;Empty cut
```

```
N60 M05 X150 Z200 ;Moving to end position
N65 M30 ; Program end
```


## Cut a 2-turn longitudinal thread (M70x1.5)

\%G33_2 (2-turn thread, thread depth 0.92 mm )
N10 G51 ; Select diameter programming
N15 T1 D1 M03 S40 ; Select tool, start spindle
N20 G00 X72 Z105
; Approach
N25 G01 X68.16 F1000
;Position at thread depth
N30 G33 Z48 K1.5
;Cut 1st thread turn
N35 G00 X72
; Retract and move
N40 Z105
; to next
N45 G01 X68.16
;start position
N50 G33 Z48 K1.5 S.OFFSET=180 ; Cut 2nd thread turn at $180^{\circ}$
N55 G00 X72 ; Retract and move
N60 M05 X150 Z200 ; to end position
N65 M30 ;Program end

## Cutting a tapered thread

\%L tapered thread
N010 N010 G33 X1 I5.0
; N010 N010 G33 Z90 X1
N020 G00 X72
N030 Z105 ;to start position
N040 M29 ; Subroutine end
\%G33
N050 G00 X0 Y0 Z0
N060 G18
N070 G51 ; Select diameter programming
N080 D1 M03 S1 ;Select tool, start spindle
N090 G00 X105 Z105 ; Start
N100 G01 X100 F1000
; Position at 1st cutting depth
N110 LL tapered thread ; Ist thread cutting
N120 M05 X150 Z200
; Move to end position
N130 M30
; Program end

### 15.2.5 Tapping (G63)

## Syntax example of tapping in $\mathbf{Z}$ direction:

G63 Z.. F.. <Spindelname>=..
(modal)

| G63 | Tapping |
| :--- | :--- |
| Z<expr> | Thread depth (target point) in the tapping axis in [mm, inch] |
| F<expr> | Feed rate in [mm/min, $\mathrm{m} / \mathrm{min}$, inch $/ \mathrm{min}]$ |
| $<$ Spindelname>=<expr> | Spindle speed consisting of spindle name according to P-CHAN-00053 and speed <br>  <br> value in [rpm] |

With this kind of tapping (G63) the position-controlled spindle is tracked by the CNC synchronously to the path motion. In this case the spindle and feed motion of the axes are adapted precisely and dynamically. A compensatory chuck is not required. The programmed feed rate must match the programmed spindle speed and the thread pitch and is calculated as follows:

Feed rate F [mm/min] = speed S [rpm] * pitch [mm/rev]
G63 is deselected by the selecting a different modal block type (e.g. linear motion G01). A nonmodal block type (e.g. dwell time with G04) does not deactivate G63.

The path feed rate (F word) and spindle speed (S word) do not necessarily need to be specified in the same NC block as G63. The feed rate calculation must always be based on the last values programmed.

An error message is output if the path feed rate or spindle speed are equal to zero with $G 63$ is selected.

M03, M04, M05, M19 cannot be programmed in combination with G331/G332.

## Attention

The spindle (or the thread tapping drill) must be at standstill when G63 is selected. This can be achieved by previously programming M05 (Stop spindle) or M19 with S.POS (Position spindle).

Cutting a left-hand thread or movement out of a thread hole is programmed with a negative $\mathbf{S}$ value.

## Programing Example

## Tapping (G63)

Tap a right-hand thread with pitch 1.25 mm , thread depth 50 mm . At a programmed spindle speed $S$ of 200 rpm the calculated feedrate is:

$$
F=200 * 1.25=250 \mathrm{~mm} / \mathrm{min}
$$

```
; ...
;...
:
```

G01 F2000 G90 X0 Y0 Z0 ; position axes
M19 S.POS=0 M3 S100 ; stop and position spindle
G63 Z-50 F250 S200 ; tap
Z0 S-200 ; retract from threaded bore
G01 F1000 X100 ; reposition, deselect tapping

## Programing Example

## Tapping (G63)

```
%Tapping G63
N05 X0 Y0 Z0
N10 G91 Z100
N20 M19 S.POS180 M3 S100 ; position spindle
N30 G63 Z-50 F300 S200 ; tap
N40 Z100 S-200 ; retract from threaded bore
N50 G01 X200 F3000 ; reposition, deselect tapping
N60 G63 Z-70 F300 S200 ; tap
N70 Z100 S-200 ; retract from threaded bore
N80 M05 G01 X300 F1000
N90 M30
```


### 15.2.6 Tapping (G331/ G332)

## Release Note

This function is available as of CNC Build V3.1.3067.01.

## Syntax example of tapping in $\mathbf{Z}$ direction:

| G331 Z.. K.. <Spindelname>=.. |  | Tapping | (modal) |
| :---: | :---: | :---: | :---: |
| G332 Z.. [ K.. ][ <SpindeIname>=.. ] |  | Withdraw thread tap | (modal) |
| G331 | Tapping |  |  |
| Z<expr> | Thread depth (target point) in the tapping axis in [mm, inch] |  |  |
| K<expr> | Thread pitch in assigned interpolation parameter in [mm/rev, inch/rev] |  |  |
| <Spindelname>=<expr> | Spindle speed consisting of spindle name according to P-CHAN-00053 and speed value in [rpm] |  |  |
| G332 | Retract from the spindle | hreaded bore (Retract) retraction. | causes |
| Z<expr> | Retract po | n of tapping axis after | in [mm, |
| K<expr> | Thread pit must be th meter is o | in assigned interpolatio ame pitch as used for nal. If not programmed | meter in aded bo tch in blo |
| <Spindelname>=<expr> | Spindle sp value in [rp applies. | consisting of spindle The parameter is opt | ccording not progr |

This type of tapping (G331/ G332) requires a position-controlled spindle which is tracked by the CNC synchronous to the path motion. In this case the spindle and the feed motion of the participating axes are matched precisely and dynamically. A compensatory chuck is not required.

The thread type is defined by specifying a sign for thread pitch.

- Pitch without or with positive sign (+): Right-hand thread, e.g. K2 or K+2
- Pitch with negative sign (-): Left-hand thread, e.g. K-2

The thread tapping axis feedrate is a product of the programmed pitch and the spindle speed. The permissible speed limits apply to the internal calculation. An error message is output if these limits are violated.

The feed rate continues to apply after tapping is completed. With the following G331/G332, the feedrate is again calculated from the related programmed or saved values of pitch and spindle speed.

G331/G332 is deselected by selecting a different modal block type (e.g. linear motion G01) and the spindles are released from the coordinated motion. A non-modal block type (e.g. dwell time with G04) does not deactivate G331/ G332.

An error message is output if the pitch or spindle speed with G331/G332 are equal to zero or the tapping axis and pitch parameters fail to match. Valid combinations are X with I, Y with J and Z with K.

M03, M04, M05, M19 cannot be programmed in combination with G331/G332.

## Attention

The spindle (or the thread tapping drill) must be at standstill when G331 is selected. This can be achieved by previously programming M05 (Stop spindle) or M19 with S.POS (Position spindle).

## Programing Example

Tapping (G331/ G332)

Tap right-hand thread with pitch 2 mm , thread depth 50 mm , spindle speed $\mathrm{S} 200 \mathrm{rpm}, \mathrm{Z}$ is tapping axis:
; ...
G01 F2000 G90 X0 Y0 Z0 ; position axes
M19 S.POS=0 M3 S100 ; stop and position spindle
; ...
G331 Z-50 K2 S200 ; tap in Z
G332 Z10 K2 S200 ; retract
G01 F1000 X50 ; reposition, deselect tapping
G331 Z-50 K2 S200 ; tap in Z
G332 Z10 K2 S400 ; retract at increased speed
G01 F1000 X100 ; reposition, deselect tapping
G331 Z-50 K2 S200 ; tap in Z
G332 Z10 ; retract, K and S from G331
G01 F1000 X150 ; reposition, deselect tapping
; ...

Tap right-hand thread with pitch 1.5 mm , thread depth 60 mm , spindle speed $\mathrm{S} 150 \mathrm{rpm}, \mathrm{X}$ is tapping axis:
; ...
G01 F2000 G90 X100 Y0 Z0 ; position axes
M19 S.POS=0 M3 S100 ; stop and position spindle
; ...
G331 X40 I1.5 S150 ; tap in X
G332 X110 I1.5 S150 ; retract
:
kernel Industrielle Steuerungstechnik GmbH

## Programing Example

## Tap at relative speed

```
%Tap at relative speed
N010 G91 G19 G0 X100 M03 S2000
N020 S2[MC_GearIn Master=S1 \ ; couple tool
RN=1 RD=1 Mode=256 \ ; spindle S2 to the
PhaseShift=1800000 WAIT SYN] ; main spindle S1
N030 #MAIN SPINDLE[S2] ; main spindle tool spindle S
N040 G331 Z-100 K1.5 S200 ; tap right-hand thread
N050 G332 Z100 K1.5 S200 ; retract from threaded bore
N060 G01 X300 F1000
N070 S2[MC_GearOut WAIT_SYN] ; release coupling to main spindle
N080 #MAIN SPINDLE[S1]
N090 M30
```


### 15.2.7 C axis machining

This functionality supplements the existing turning functions and permits the face and lateral surface machining of cylindrical workpieces on lathes and milling machines with revolving base. The workpiece is moved by the rotary axis or spindle ( C axis) and the driven tool (e.g. milling cutter) by the two translatory axes X (or Y ) and Z . Settings are required in the parameters P-CHAN-00008 and P-AXIS-00015 for C axis machining.
Face and lateral surface machining can be described in Cartesian coordinates.
All interpolation types (such as linear, circular or spline interpolation) are supported on the end face and lateral surface. The functionality also permits the machining of path contours running through the turning centre point. The C axis is automatically aligned on lathes.
The 2.5D tool radius compensation can be used with the familiar G commands.
The use of extended dynamic monitoring can specifically prevent dynamic axis characteristics from being exceeded with the C axis function and also with contours running close to the turning centre point.

The main axes for all machining modes are $\mathrm{X}, \mathrm{Y}$ (depending on machine type), Z and C .

### 15.2.7.1 Exchange spindles in coordinated motion (\# CAX, \#CAX OFF)

This "basic mode" is required in particular for C axis machining on lathes because in this case the position-controlled spindle has to be converted into a rotary path axis (e.g "C").

Notice
C axis machining is also possible on milling machines or machining centres which are designed with rotary workpiece fixtures (e.g. revolving base). In this case, it is not necessary to select \#CAX.

The three physical axes $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ and the C axis replaced in coordinated motion can be directly programmed. Linear axes are programmed in Cartesian coordinates and the $C$ axis in angle units.

Radius and diameter programming depends on G52/G51.
The two linear axes define the main plane.
ZX (G18) or YZ (G19).

```
#CAX [ [ [<main_spindle_name>, ] <C_axis_name> ] ]
```

<main_spindle_name>
<C_axis_name>

Only the main spindle name can be programmed according to P-CHAN-00053. If a spindle other than the C axis is used, it must first be declared as the main spindle (see programming example in Section [PROG// Changing the main spindle [ $>685]]$ ). Otherwise, an error message is output.
Freely definable name of the C axis in the NC program. If no C axis name is programmed, the default name from P-CHAN-00010 is used.

The main plane (circular interpolation, tool radius compensation, etc.) remains the same as before activation of the C axis.

An error is generated if a command for this spindle (M3, M4, M5 [ 624], etc.) is programmed although the axis is still declared as a C axis in the coordinated motion.

The $C$ axis is deselected, i.e. the axis is released to the spindle interpolator, by the following:

## \#CAX OFF

## Programing Example

C axis machining
Exchange spindles in coordinated motion

```
; ...
#CAX ; Assuming: default C axis is "C"
G01 G90 X50 Z10 C90 F200
#CAX OFF
; ...
#CAX[S, C] or #CAX[C] ; Assuming: main spindle is "S"
G01 G90 X50 Z10 C90 F200
#CAX OFF
```

```
; ...
#MAIN SPINDLE [S2] ; "S2" becomes new main spindle "S"
#CAX[S, C]
G01 G90 X50 Z10 C90 F200
; ...
#CAX OFF ; deselect C axis mode
; ...
#CAX[S3, C]; Error, "S3" is not the main spindle
```


### 15.2.7.2 Face machining (\#FACE, \#FACE OFF)

This mode is selected for lathes and machining centres. The desired contour on the face is programmed in millimetres or inches using a virtual Cartesian coordinate system.


Fig. 161: Face machining

## Notice

As of CNC Build V3.00 it is imperative to assign the parameter P-CHAN-00262 with the kinematic ID used, depending on P-CHAN-00008, in order to use face machining.
a) For face transformation 1 with P-CHAN-00008=1 - ID 13
b) For face transformation 2 with P-CHAN-00008=2 - ID 14

The three logical axes $X, Y$ (or $C$ ) and $Z$ are provided to program the contour on the face in Cartesian coordinates.


Fig. 162: Front view of face machining process

The figure below shows each of the main planes in face machining. Only the G17 plane is of technological importance.


Fig. 163: Main places of face machining
\#FACE [ <name of 1st main axis>, <name of 2nd main axis> ]
<name of 1st main axis>
<name of 2nd main axis>

Name of the first main axis according to the current main plane.
Name of the second main axis according to the current main plane (virtual Cartesian axis).

When selected. the main plane (circular interpolation, tool radius compensation, etc.) is always defined by the 1st and 2nd main axes. It is not permitted to change the main plane with G18, G19 while face machining is active.

Notice
Programmed tracking axes are not affected by the transformation.

This mode is deselected by:

## \#FACE OFF

The above command returns to the last active mode (e.g. mode 1). This means that the last active main plane is selected automatically and the last active axis offsets are restored.

## Programing Example

Programming example for lathes

Example with axis name "C" for second main axis. main axis

```
; ..
#CAX[S, C] ; Assuming main spindle is "S"
#FACE[X, C] ; select face machining
; ...
G01 X40 C-30 Z50 F1000 ; pre-position
G01 Z30 ; approach
G01 X10 C40 ; travel contour
G01 Z50 ; retract
; ..
#FACE OFF
#CAX OFF
; ..
M30
```

Example with axis name " $Y$ " for second main axis. main axis.
Note: No other axis with the identical name " Y " may exist in NC channel.

```
; ...
#CAX[S, Y] ; Assuming main spindle is "S"
#FACE[X, Y] ; select face machining
; ..
G01 X40 Y-30 Z50 F1000 ; pre-position
G01 Z30 ; approach
G01 X10 Y40 ; travel contour
G01 Z50 ; retract
; ..
#FACE OFF
#CAX OFF
; ..
M30
```


## Programing Example

## Programming example for machining centres

The rotary axis (workpiece axis) in the channel is "C2". It is not necessary to program the \#CAX command.

```
; ...
#FACE[X, C2] ; select face machining
;..
G01 X40 C2=-30 Z50 F1000 ; pre-position
G01 Z30 ; approach
G01 X10 C2=40 ; travel contour
G01 Z50 ; retract
; ..
#FACE OFF
;...
M30
```


### 15.2.7.3 Surface machining (\#CYL, \#CYL OFF)

This mode can be selected for lathes and machining centres. The desired contour on the cylindrical surface is programmed in millimetres (or inches) using a virtual coordinate system.


Fig. 164: Lateral surface machining

## Notice

As of CNC Build V3.00 it is imperative to assign the parameter P-CHAN-00262 with the value 15 for this transformation in order to use lateral surface machining.

The three logical axes $X, Y, Z$ are provided to program the contour on the lateral surface in Cartesian coordinates. In this mode, the reference radius R of the workpiece must also be programmed.


The main plane in lateral surface machining is formed by Z-C.

\#CYL [ <1st main_axis_name>, <2nd main_axis_name>, <3rd main_axis><expr>] (modal)
<1st main_axis_name>
<2nd main_axis_name>
<3rd main_axis_name><expr>

Name of the first main axis according to the current main plane.
Name of the second main axis according to the current main plane (virtual linear axis, development).
Axis name of the third main axis according to the current main plane with specification of the reference radius in [mm, inch].

When selected. the main plane (circular interpolation, tool radius compensation, etc.) is always defined by the 1st and 2nd main axes. It is not permitted to change the main plane by G18, G19 while lateral surface machining is active.

This mode is deselected by:

## \#CYL OFF

The above command returns to the last active mode (e.g. mode 1). This means that the last active main plane is selected automatically and the last active axis offsets are restored.

## Programing Example

Programming example for lathes

Example with axis name "C" for second main axis. main axis

```
; ..
G01 C100 F500
G02 Z100 R50
G01 C0
ZO
; ..
#CYL OFF
#CAX OFF
M30
```

\#CAX [S, C] ; Assuming " $S$ " is main spindle
G01 X60 C45 ; approach and positioning movement; X:60mm C:45
\#CYL [Z, C, X60] ; select lateral surface machining
G00 G90 Z0 C0 ; Z: 0mm C:0mm!

### 15.2.7.4 Switching between face and lateral surface machining

All C axis modes are deselected normally using the deselection commands described (for example using the \#CYL OFF command). It is also permitted to change directly to a different machining mode, e.g. between face and lateral surface machining, without previously deselecting the current active mode. The programming example below shows a typical NC sequence for changing between C axis modes:

## Programing Example

Switch between C axis modes

```
N10 #CAX [..] ;Adopt the LR spindle in the coordinated motion
; . . . . . . . . . . . .
N120 #FACE [..] ;Select face machining
; . . ...........
N230 #CYL [..] ;Direct transition to lateral surface machining
N300 #CYL OFF ;Deselect lateral surface machining and transition
;to conven. machining with physical C axis
N400 #CAX OFF ; Return the C axis to the position-controlled spindle
N500 M30
```


### 15.2.7.5 Tool offsets

The commands \#FACE and \#CYL result in the implicit selection of kinematics. For this reason, neither a kinematic ID requires a \#KIN ID [> 720] [..] nor transformation activation with \#TRAFO ON [> 713].

## Tool offsets for face machining

Face machining supports 2 machine types (lathe/milling machine). The corresponding tool offsets must be entered in the channel parameters in the assigned offset data of the kinematic IDs 13 and 14. Alternatively, this can also be executed in the tool data.


Fig. 165: Tool offsets for face machining

## Example

## Examples of entries in channel parameters

## for CNC Builds as of V3.00

\# FACE[], Face machining on a lathe (KIN-ID 13):

```
trafo[0].id
trafo[0].param[0]
trafo[0].param[2]
```

$1080000 \quad \mathrm{Z}$ offset [0.1 mm ]
trafo[0].param[1] $0 \quad$ C angular offset [10 $0^{-40}$ ]
13
900000 X offset [0.1 $\mu \mathrm{m}$ ]
\# FACE[], Face machining on a milling machine (KIN-ID 14):

```
trafo[1].id
trafo[0].param[0]
trafo[0].param[1]
trafo[1].param[2]
14
1080000 Z offset [0.1\mum]
0 C angular offset [10-40}
900000 X offset [0.1\mum]
```

-rafo[1].param[2]

## CNC Builds < V3.00

\# FACE[], Face machining on a lathe (KIN-ID 13):

| kinematik[13].param[0] | 1080000 | Z offset [0.1 $\mu \mathrm{m}]$ |
| :--- | :---: | :--- |
| kinematik[13].param[1] | 0 | C angular offset [10-40] |
| kinematik[13].param[2] | 900000 | X offset [0.1 $\mu \mathrm{m}]$ |

\# FACE[], Face machining on a milling machine (KIN-ID 14):

| kinematik[14].param[0] | 1080000 |  | offset [0.1 ${ }^{\text {ma }}$ |  |
| :---: | :---: | :---: | :---: | :---: |
| kinematik[14].param[1] | 0 | C | angular offset | $\left[10^{-40}\right]$ |
| kinematik[14].param[2] | 900000 |  | offset [0.1 ${ }^{\text {m }}$ ] |  |

## Tool offsets for lateral surface machining

Lateral surface machining implicitly executes the selection of the kinematic with ID 15.


Fig. 166: Tool offsets for lateral surface machining
\#CYCL[], Lateral surface machining lathe (KIN-ID 15):
CNC Builds as of V3.00

```
trafo[0].id
trafo[0].param[0]
trafo[0].param[0]
trafo[2].param[0]
```

15
$700000 \quad \mathrm{X}$ offset [0.1 mm ]
$0 \quad C$ angular offset $\left[10^{-40}\right]$
$1200000 \quad Z$ offset [0.1 $\mu \mathrm{m}]$

CNC Builds < V3.00


### 15.2.8 Gear change (G112)

G112 <spindle_name><expr> (non-modal)

## G112

<Spindle_name><expr>

## Gear change

Gear data record consisting of the spindle name as described in the decoder parameter list and number of the data record

As opposed to gear changing via M40-45 in which the mechanical gear change operation is also performed implicitly, programming G112 together with the S word only triggers the update of the spindle gear data (dynamic values) of one step.
The user must explicitly program the mechanical changing of the correct gear stage, e.g. by selfdefined M functions or as mentioned above using M40-45.

## Attention

The spindle must be at standstill before gear data record is changed. This can be achieved in the previous NC block by programming a spindle stop (M5) or a spindle positioning (M19...)

## Programing Example

Gear change (G112)
\%Test_G112
N010 G112 S2 (Load dynamic data of gear stage 2 for spindle "S")
N020 M3 S9000
N030 M5 (Stop spindle)
N040 G112 S1 (Load dynamic data of gear stage 1 for spindle "S")
N050 M4 S8000
N060 M30

### 15.2.9 Homing in DIN syntax( G74)

G74 <Spindelname><dummy_expr>
(non-modal)

```
G74
<spindle_name><dummy_expr>
```

Homing
Spindle name according to P-CHAN-00053 with specification of a value.

Homing can be conducted for closed-loop position-controlled spindles. As opposed to referencing linear axes, the values programmed with the spindle name have no significance relating the referencing sequence and are only required to represent a complete syntax.
It is not permitted to program spindle M functions in the same NC block as G74.

A distinction must be made in the following cases when referencing spindles:

## Programing Example

## Homing (G74)

```
Case 1:
Spindle referencing starts simultaneously with
Y axis referencing:
    Nxx G74 X2 Y1 S1
```


## Case 2:

```
Same as 1.! The system continues to the next NC block without waiting until the spindle is referenced so that the \(X\) axis is referenced quasi simultaneously:
    Nxx G74 S1
    NYY G74 X1 Y2
```


## Case 3:

```
Axes \(X\) and \(Y\) are first referenced. Spindle referencing then starts:
Nxx G74 X1 Y2
Nyy G74 S1
```


### 15.2.10 Spindle override in DIN syntax (G167)

G167 <spindle_name><expr> (non-modal)

## G167

<spindle_name><expr>

Set spindle override to 100\%
Spindle speed consisting of spindle name according to P-CHAN-00053 and specification of a value. The value has no significance but is only required to represent a complete syntax.

The G167 function deactivates external influencing of spindle override and implements the speed actually programmed.

## Programing Example

## Override (G167)

### 15.3 Programming in spindle-specific syntax

Spindle-specific syntax offers the advantage that several spindles can be programmed mutually independent in the same NC block.

This is carried out within a bracketed expression attached to the spindle name. Only specific commands are permitted in this bracketed expression and these commands are always handled and executed spindle-specific. The main spindle can only be programmed by its main spindle name (P-CHAN-00053).

```
<Spindelname> [ [ M3 | M4 | M5 | M19 ] [ REV.. ] [ POS.. ] { M.. } { H.. }
    [ G74 ] [ G167] [CALLAX | PUTAX ] [ GET_DYNAMIC_DATA ] [G130]
    [ [G135 | G137] [G136..]] [ FEED_LINK...] [ OVERRIDE.. ] { \ }]
    {<SpindeIname> [....] }
```

<Spindelname>
M3, M4, M5, M19
REV<expr>
POS<expr>
M<expr>
H<expr>
G74
G167
CALLAX
PUTAX
GET_DYNAMIK_DATA
G130
G135, G136<expr>, G137
FEED_LINK...
OVERRIDE...
1

Spindle name according to [1] [ 819]-3 and P-CHAN-00053
Spindle M functions
Spindle speed
Spindle position
User-specific M functions
User-specific H functions
Homing
Spindle override 100\%
Call spindle axis
Release spindle axis
Adopt new tool dynamic data
Acceleration weighting
Feedforward control
Spindle feed link
Spindle override
Separator ("backslash") for clear programming of the command over multiple lines.

## Programing Example

## Programming in spindle-specific syntax

```
:
N10 S[M3 REV500 M19 POS45 M18 M15 H20 ...] S2[M4 REV5000]
Nxx
:
```


### 15.3.1 $\quad$ The spindle $M$ functions

### 15.3.1.1 Moving spindle in spindle-specific syntax ((M3/M4/M5)

| M03 | Spindle rotation clockwise (cw) | (modal) |
| :--- | :--- | :---: |
| M04 | Spindle rotation counter-clockwise (ccw) | (modal) |
| M05 | Stop spindle | (modal) |

The spindle M functions M03 and M04 define the spindle direction of rotation and must be used in conjunction with the spindle speed (REV word). M05 stops spindle rotation. Note that this spindle $M$ function is the default spindle mode after controller start-up and initial program start. These M functions are modal and may only be programmed on their own within the bracketed expression. Spindle rotation is activated if M03 or M04 are programmed and a valid speed (REV) is set. If no M05 is set at program end, the spindle continues to rotate.

## Programing Example

Programming of one spindle " S ":

```
N10 S[REV1000] (Speed 1000 rpm is stored,)
(no spindle rotation because M05 is default)
N20 S[M03] (Spindle rotation cw at 1000 rpm)
N30 S[M04] (Spindle rotation ccw at 1000 rpm)
N40 S[REV500] (Spindle rotation ccw at 500 rpm)
N50 S[M05 REV300] (Spindle stop, speed 300 rpm is stored)
N60 S[M04] (Spindle rotation ccw at 300 rpm)
N70 S[M05] (Spindle stop)
N80 S[M03 REV1000] (Spindle rotation cw at 1000 rpm)
N90 M30 (Program end)
```


## Programing Example

## Programming two spindles "S2"and "S2":

| N10 S[M03 REV1000] | S2 [M04 REV2000] | (S CW $1000 \mathrm{rpm}, \mathrm{S} 2 \mathrm{ccw} 2000 \mathrm{rpm})$ |
| :---: | :---: | :---: |
| N20 S[M05] | S2 [REV1500] | (S stop, S2 ccw at 1500 rpm ) |
| N30 S [M04] |  | (S ccw 1000 rpm ) |
| N40 | S2 [M05] | (S2 stop) |
| N50 S [M05 REV300] |  | (S stop, store speed 300 rpm ) |
| N60 S [M04] | S2 [M04] | (S ccw $300 \mathrm{rpm}, \mathrm{S} 2 \mathrm{ccw} 1500 \mathrm{rpm}$ ) |
| N70 S [M05] | S2 [M05] | (S stop, S2 stop) |
| N80 M30 |  | (Program end) |

Channel parameter list [1] [> 819]:
The synchronisation modes must be defined spindle-specific for M3, M4, M5. The M function is not executed for synchronisation mode "0" (NO_SYNCH).
:
spindel[0].bezeichnung
spindel[0].log_achs_nr
spindel[0].s synch
spindel[0].m $\overline{3}$ synch
spindel[0].m4_synch
spindel[0].m5_synch
spindel[0].m1 $\overline{9}$ _synch

S1
6
0x00000001
$0 \times 00000002$
$0 \times 00000002$
$0 \times 00000008$
$0 \times 00000001$

### 15.3.1.2 Positioning spindle in spindle-specific syntax (M19, POS)

| M19 | Position spindle | (non modal) |
| :--- | :--- | :---: |
| POS | Spindle position | (modal) |

The spindle M function M19 executes spindle positioning and must be used in conjunction with the POS word. M03/M04 and the spindle speed (REV) in the same NC block are optional. However, a valid spindle speed (>0) must be set. M19 may not be used together with M5 (spindle stop).

Spindle position POS in [ ${ }^{\circ}$ ] is modal and need not be respecified if M19 is programmed again. If no spindle position was previously programmed, the motion is moved to position "zero" by default. If the spindle is not rotating, positioning is executed with the shortest motion path. Spindle positioning with M19 is only permitted for position-controlled spindles (closed loop).

## Programing Example

## Programming of one spindle "S":

```
N10 S[REV100 POS45 M19 M3] (Move cw at 100 rpm to position 45)
N20 S[M03 REV1000] (Spindle rotation cw at 1000 rpm)
N30 S[M19 M4 REV150]
N40 S[M05]
N50 S[M19]
N60 S[REV200 M19 POS90]
N70 S[M03 REV1000]
N80 S[M05]
N90 M30
    (Move ccw at 150 rpm to position 45)
    (->POS is modal!)
    (Spindle stop)
    (Move on shortest path at }150\mathrm{ rpm to)
    (position 45)
    (Move on shortest path at 200 rpm to)
    (position 90)
    (Spindle rotation cw at 1000 rpm)
    (Spindle stop)
    (Program end)
```


## Programing Example

## Programming two spindles "S2"and "S2":

```
(For N10: Move both spindles cw at 1000 rpm to position 45)
N10 S[M03 REV200 M19 POS45] S2[M04 REV200 POS45 M19]
N20 S[REV1000] S2[REV1500] (S E FAC 1000 rpm)
    (S2 ccw at 1500 rpm)
N30 S[M04] (S ccw 1000 rpm)
N40 S2[M19 POS0]
    (Move S2 ccw to position 0)
N50 S[M05 REV300]
    (S stop, store speed 300 rpm)
(For N60: Move S Cw at 300 rpm to position 90,)
(For N60: Move S2 cw at 200 rpm to position 45)
N60 S[M03 M19 POS90] S2[M03 REV200 POS45 M19]
N80 M30 (Program end)
Channel parameter list [1] [> 819]:
```

The synchronisation mode must be defined spindle-specific for M 19 . The M function is not executed for synchronisation mode "0" (NO_SYNCH).
:

| spindel[0].bezeichnung | S1 |
| :--- | :--- |
| spindel [0].log_achs_nr | 6 |
| spindel[0].s_synch | $0 \times 00000001$ |
| spindel[0].m3_synch | $0 \times 00000002$ |
| spindel[0].m4-synch | $0 \times 00000002$ |
| spindel[0].m5-synch | $0 \times 00000008$ |
| spindel[0].m19_synch | $0 \times 00000001$ |

### 15.3.2 Spindle speed (REV)

Values can be assigned directly or by means of parameters to the REV word and decimal numbers are also permitted (REAL format).
A distinction must be made between the following types of use in conjunction with the spindle $M$ functions in the case of the REV word:

1. REV word in conjunction with $\mathrm{M} 03 / \mathrm{M} 04$ or M 19 :

If the REV word is programmed in conjunction with M03/M04 or M19, the value following the REV word is interpreted as the spindle speed and is output to the spindle.
2. REV word in conjunction with M05:

Together with M05, the value following the REV word is transferred to the working data as the spindle speed but is not output to the spindle
The REV word on its own does not generate a motion in the NC program. This requires that a spindle mode M03, M04, M19 is known. Accordingly, programming of M03, M04 and M19 only results in a motion if the REV word is set.

## Notice

An error message is output if the REV value is negative.

## Programing Example

Programming with spindle S1:

```
N10 S1[REV300] (Speed 300 rpm is stored)
N20 S1[M04]
N30 S1[M03 REV1000]
N40 S1[REV500]
N40 S1[REV500]
N60 S1[M04]
N70 S1[M05] (Spindle stop)
N80 M30
N20 S1[M04]
(Spindle rotation ccw at 300 rpm)
(Spindle rotation cw at 1000 rpm)
(M03 active, therefore spindle rotation cw at 500 rpm)
N (Spindle stop, speed 100 rpm is stored)
(Spindle rotation ccw at 100 rpm)
```


### 15.3.3 User-specific M/H function in spindle-specific syntax

All user-specific M/H functions (technology information) programmed within the bracket expression are always handled and output spindle-specific.

## Programing Example

Program one spindle "S2" with two user-specific M functions:

```
N10 S2[M3 REV300 M10 M11] (S2 cw rotates at 300 rpm and outputs M10/M11)
N20 M30 (Program end)
```


## Programing Example

Program two spindles "S" and S2:

```
N10 S[M10 M11] S2[REV1000 M3 M11] (S executes M10 and M11, S2 rotates)
    (at 1000 rpm and outputs M11)
    (Program end)
```

Channel parameter list [1] [> 819]:
The user-specific M/H functions are defined in P-CHAN-00027 and P-CHAN-00041. The M/H function is not executed in the case of synchronisation mode "0" (NO_SYNCH).
:
\# Definition of $M$ functions and synchronisation types
:
\# Definition of $M$ functions and synchronisation types
m_synch[1] 0x00000001 MOS
m_synch[2] 0x00000002 MVS_SVS
:
m_synch[10] 0x00000002 MVS_SVS
m_synch[11] 0x00000008 MVS_SVS
m_synch[12] 0x00000004 MVS_SVS
m_synch[48] 0x00000008 MNS_SNS
m_synch[49] 0x00000002 MVS_SVS
: : :
\# Definition of $H$ functions and synchronisation types
h_synch[1] 0x00000001 MOS
h_synch[2] 0x00000001 MOS
:

### 15.3.4 Homing in spindle-specific syntax (G74)

Homing can be executed spindle-specific. Programming the spindle $M$ functions is not permitted together with G74.

## Programing Example

Programming two spindles "S2"and "S2":

```
N10 S[G74] S2[G74] ;Homing S and S2
N20 M30 ;Program end
```


### 15.3.5 Spindle override in spindle-specific syntax (G167)

The G167 function deactivates external influencing of spindle override spindle-specific and executes the actually programmed rotational speed. The effect of the programmed override value for one spindle [ $>683$ ] is retained.

## Programing Example

N10 S2[M3 REV1000]
N20 S2[G167 REV1000]

N30 S2[REV3000]
N40 M30

```
(With override 50% speed is 500 rpm)
(Override influencing off,)
(spindle rotates at 1000 rpm)
(Override influencing active again,)
(speed 1500 rpm)
(End of program)
```


### 15.3.6 Releasing/requesting spindle axes (PUTAX/CALLAX)

## Release Note

The availability of this function depends on the configuration and the scope of the version.

These commands can be used to program the release or call of the spindle axis spindle-specific. The commands may not be used simultaneously with other spindle-specific commands.

## Programing Example

```
%s-putcallax
(Move axis as spindle)
N10 S[CALLAX]
N20 M03 S1000
N30 G04 X2
N40 M05
(Exchange axis from spindle interpolator in channel)
N50 S[PUTAX]
N60 #CALL AX[ C, 4, 3]
(Drive axis in channel)
N70 X10 Y20 Z30 C40
N80 X-10 Y-20 Z-30 C--40
(Exchange axis from channel in spindle interpolator)
N90 #PUT AX[C]
N100 S[CALLAX]
(Move axis again as spindle)
N110 M04 S1000
N120 G04 X2
N130 M05
N140 M30
```


## Programing Example

## Programming multiple spindles:

```
:
N10 S[CALLAX] S2[CALLAX] S3[PUTAX] (S, S2 call their axes,
    (S3 releases its axis)
    (S rotates cw at 200 rpm),
    (S2 is homing)
N30 S3[ M4 REV3000] (Error, S3 currently has no axis)
N40 S[PUTAX REV400) (Error, PUTAX may not be programmed alone)
```


### 15.3.7 Adopt tool dynamic data (GET_DYNAMIC_DATA/ DEFAULT_DYNAMIC_DATA)

## Release Note

The availability of this function depends on the configuration and the scope of the version.

The tool dynamic data (minimum/maximum speed, max. acceleration) takes effect automatically on transition of the spindle from standstill to interpolation after programming a new tool (D word, \#TOOL DATA). Changed tool dynamic data is adopted and considered for a rotating spindle by the spindle-specific command "GET_DYNAMIC_DATA".

When a spindle is at standstill, a change back to tool-independent default dynamic data can be executed by the command "DEFAULT_DYNAMIC_DATA". The currently active gear stage (P-TOOL-00016/P-TOOL-00017) is not changed.

These commands may not be used simultaneously or in combination with other spindle-specific commands.

## Programing Example

## Adopt dynamic tool data

```
N10 T1 D1 ;Supply tool dynamic data to the
    ;tool data
N15 M6 ;Change of tool 1
N20 S[M3 REV2002] ;S rotates at 2000 rpm with tool dynamic data
    ;from D1
N25 X100 Y100 ;Motion block with tool dynamic data D1
N30 T99 D99 ;Supply tool dynamic data to the
    ;tool data while spindle rotates (N20)
N35 X200 Y150 ;Motion block with tool dynamic data D1
N40 S[GET DYNAMIC DATA] ;Adopt tool dynamic data D99
N45 X150 Y200 ;Motion block with tool dynamic data D99
N50 X50 Y50 ;Motion block with tool dynamic data D99
N60 S[M05] Z100 ;Spindle stop
N70 S[DEFAULT DYNAMIC_DATA] ;Revert to default dynamic data
N99 M30 ;Program end
```


### 15.3.8 Commanding spindle feedforward control (G135/G136/G137)

## Release Note

The availability of this function depends on the configuration and on the version scope.

These commands permit the spindle-specific programming of feedforward control. The commands may not be used simultaneously with other spindle-specific commands.
Activation is programmed with G135.
Spindle-specific, percentage weighting of the calculated feedforward control variables takes place with G136. It is limited to $100 \%$.

G137 deactivates feedforward control. It is also possible to specify the selection and weighting of the feedforward control in the same block.

If feedforward control is disabled or enabled during the NC program, the weighting factors remain at the values set by G136 or, if no G136 is programmed, to $100 \%$.

## Programing Example

## Commanding spindle feedforward control

```
S [G135]
S[G136=80]
```

;Activate feedforward control for $S$
; Define weighting in percent
; Deactivate feedforward control
S2[G135 G136=90] ;Activate at 90\% weighting for S2
S2[G136=0] ;Change weighting to 0\%
S1 G135] ;Activate at 100\% default weighting for S1

### 15.3.9 Spindle feed link (FEED_LINK)

Normally, the speed of a spindle is only controlled by the program. In general, there are no other options to influence spindle speed depending on other parameters. For some special technologies (e.g. HSC milling, edge banding in wood machining) and to obtain a uniform milling profile on the workpiece surface, it is necessary to influence spindle speed by external variables.

The following command can link the spindle-specific speed dynamically to path feed by specifying various parameters. When set accordingly, the spindle speed then adapts automatically to various feedrates. This is necessary in particular for materials which may be damaged by unsuitable cutting parameters (e.g. scorch marks on wooden workpieces or melted plastic etc.).

## Syntax for programming a spindle feed link:

<Spindelname> [ FEED_LINK [ ON | OFF ] [ [ [ FACT=.. ] [ CORR=.. ] ]| AUTO] [ MAIN ] SRC=.. ]
<spindle name>
FEED_LINK
ON
OFF
FACT=<expr>

CORR=<expr> Correction factor in [0.1\%] to modify the link factor. The resulting link factor then results from the product of FACT and CORR (CRES = FACT*CORR).
AUTO Flying link: The link factor is calculated automatically from the current commanded spindle speed and the current path feed and it remains constant until the link is deselected.
Spindle feed link is only effective if at least one main axis is involved in motion. The spindle speed is not influenced if only tracking axes are moved.
Identifier of feed link source:

- FEED_VEL - path feed (default)
- EDGE_VEL - edge velocity for edge banding
- The keyword AUTO can only be programmed in combination with ON. FACT and CORR can even be reprogrammed while a feed link is active.
- The keyword MAIN can only be programmed in combination with ON.
- When the link is selected, the spindle must be in the state of endless rotation at a sped of != 0 . The path feed must also be != 0 when the link factor is defined automatically.
- The spindle speed which is active when the feed link is deselected is retained as the commanded spindle speed. If necessary, the spindle speed must be redefined using the $S$ word after the feed link is deselected.


## Attention

A spindle command via the $S$ word has no effect when the feed link is active.
$\square$

## Programing Example

## Example 1

(Manually programmed link factor, the effective link factor is changed by programming the correction factor.

```
%feed_link1
N10 G00 X0 Y0 Z0
N20 G01 G90 X20 F2
N30 M03 S15
N40 G01 G90 X40
N50 S[FEED LINK ON FACT=500]
N60 G01 G91 X5 F2
N70 S[FEED_LINK CORR=1100]
N80 X10 F2
N90 S[FEED LINK CORR=900]
N100 X20 F2
N110 S[FEED_LINK CORR=1100]
N120 X40 F2
N130 S[FEED LINK CORR=900]
N140 X80 F2
N150 S[FEED_LINK CORR=1100]
N160 X40 F2
N170 S[FEED_LINK CORR=900]
N180 X20 F2
N190 S[FEED_LINK CORR=1100]
N200 X5 F2
N210 S[FEED_LINK OFF]
N220 M30
```


## Programing Example

Example 2
(Link factor is generated automatically from the current commanded spindle speed and the current path feed).

```
%feed_link2
N10 G00 X0 Y0 Z0
N20 G01 G90 X20 F1
```

```
N30 M03 S15
N40 G01 G90 X40
N50 S[FEED LINK ON AUTO]
N60 G01 G91 X5 F1
N70 X10 F2
N80 X20 F4
N90 X40 F8
N100 X80 F16
N110 X40 F8
N120 X20 F4
N130 X10 F2
N140 X5 F1
N150 S[FEED_LINK OFF]
N160 M30
```


## Programing Example

## Example 3

(Link factor is generated automatically, feed link only acts with main axis motions).

```
%feed_link3
N10 M03 S200
N20 G01 X0 Y0 Z0 C0 F10
N30 S[FEED LINK ON AUTO MAIN]
N40 G01 X50
N50 G01 X70 Y30
N60 G01 C45 ;Spindle speed remains constant
N70 G01 Z40 F15
N80 G01 Z60
N90 G01 C90 ;Spindle speed remains constant
N100 G01 C120 ; Spindle speed remains constant
N110 G01 X50 Y50 Z50
N120 S[FEED_LINK OFF]
N130 M03 S200
N140 G01 X10
N150 M05 S0
N160 M30
```


### 15.3.10 Programmable spindle override

This function influences spindle speed in NC program. The spindle-specific programmed override is active when the assigned spindle moves.
If an external override is also defined, the effective override results from the multiplication of the two override values.

## Notice

The G167 [ 676] function only suppresses the effect of the external override value.

## <Spindelname> [ OVERRIDE SPEED_FACT=..]

<Spindelname>
OVERRIDE

SPEED_FACT=<expr>

Name of the spindle
Identifier for spindle-specific override programming. Must always be programmed as the first keyword.
Override factor for spindle speed blocks [0.1\%-200\%]

## Programing Example

## Programmable spindle override

```
%spdl_override
N10 G01 X100 Y100 Z100 F1000
N20 S[M3 REV1000]
N40 S[OVERRIDE SPEED_FACT=20] ; Spindle override 20%
N50 X0
N60 Y0 ;G01 motion with S200
N70 Z0 ;G01 motion with S200
N60 S[OVERRIDE SPEED_FACT=100] ; Spindle override 100%
N90 Y100 ;G01 motion with S1000
N100 Z100
N110 X200 Y200
N120 X300 Y300 Z200 ;G01 motion with S1000
N130 M5
M30
```


### 15.3.11 Acceleration weighting (G130)

The G130 function can change the acceleration of the spindle axis.
Acceleration can be influenced by a percentage change in the associated acceleration characteristic values. With a jerk-limited profile, these values are the axis parameters P-AXIS-00001 and P-AXIS-00002.

If programming takes place with G130, all axes which are not programmed or not yet programmed are set to $100 \%$. Every time these functions are selected, the $100 \%$ weighting is taken irrespective of previous programming.

Therefore, $50 \%$ programmed twice in succession means the setting is made to $50 \%$ and not to $25 \%$. A weighting of over $100 \%$ is possible up to maximum axis acceleration P-AXIS-00008.

## Attention

At program end, the G130 weighing factor is restored to $100 \%$.

## Programing Example

Acceleration weighting (G130)

```
N10 S[G130=70] ;Spindle acceleration is limited to 70%
N20 M03 S1000 ; Endless rotation CW
N30 S[G130=60] ;Spindle acceleration is limited to 60%
N40 M04 S1000 ; Endless rotation CCW
```


### 15.4 Changing the main spindle (\#MAIN SPINDLE)

\#MAIN SPINDLE [ [ <Spindelname> | <SpindeInummer> ] ]
<Spindelname>
<Spindelnummer>

Default spindle name according to [1] [ 819]-3.
log. axis number of the spindle according to [1] [ $>819]-3$.

The \#MAIN SPINDLE command can be used to change the definition of the main spindle in the NC program. The new main spindle is selected by specifying the default name (P-CHAN-00053) or the related logical axis number.
The initial state (as after start-up) can be restored without programming a spindle name, i.e. the spindle preset in the channel parameter list P-CHAN-00051 becomes the main spindle again.

Channel parameter list [1] [> 819]:
Configuration example of a 1-channel system with 3 spindles. Spindle with axis number 6 is the main spindle:

```
# Spindle data
# =============
spdl_anzahl 3
main_spindle_ax_nr 6
main_spindle_name S
spindel[0].bezeichnung S1
spindel[0].log_achs_nr 6
spindel[1].bezeichnung S2
spindel[1].log_achs_nr 11
spindel[2].bezeichnung S3
spindel[2].log_achs_nr 30
```


## Configuration after start-up:

S 1 is the main spindle with the name "S".
"S2" and "S3" are other spindles.

## Programing Example

## Changing the main spindle

## \%

N10 S100 M3 S2[REV200 M3] S3[REV300 M4]
N20 \#MAIN SPINDLE [S2] (S2 is new main spindle "S")
N30 S110 M3 S1[REV210 M3] S3[REV310 M4]
N40 \#MAIN SPINDLE [S3] (S3 is new main spindle "S")
N50 S120 M3 S1[REV220 M3] S2[REV320 M4]
N60 \#MAIN SPINDLE (Back to initial state S1 -> "S")
N70 S150 M3 S2[REV250 M3] S3[REV350 M4]
N80 M5 S2[M5] S3[M5] (All spindles STOP)
N99 M30

## Notice

As long as a spindle is a main spindle, it can either be programmed with the defined main spindle name P-CHAN-00053 or with its default name [1] [ 819]-3. It can only be addressed exclusively by its default name again after another main spindle is selected with \#MAIN spindle[ ].

The following applies in the example above:

| Permissible names | Spindle 1 | Spindle 2 | Spindle 3 |
| :---: | :---: | :---: | :---: |
| $\ldots$...after start-up | S or S1 | S2 | S3 |
| $\ldots$ after \#MAIN SPINDLE [S2] | S1 | S or S2 | S3 |
| $\ldots$ after \#MAIN SPINDLE [S3] | S1 | S2 | S or S3 |
| $\ldots$ after \#MAIN SPINDLE | S or S1 | S2 | S 3 |

As already mentioned, the main spindle can be programmed in the conventional DIN syntax. In this case, all commands in the table in Section Spindle programming [ [817] can be used. The main spindle may also be programmed in spindle-specific syntax. However, in this case only the restricted command set then is available (see also the table in Section Spindle programming. [ 617].

## Programing Example

The following NC lines are also permissible for the main spindle:

```
:
N10 S1=1000 M3 or
N20 S1000 M3 or
N30 S1[REV1000 M3] or
N40 S[REV1000 M3]
:
```


### 15.5 Synchronous spindle operation

Besides the synchronous mode of path axes (definition, activation, deactivation), the LINK command can also be used to define master/slave relationships for spindle axes.

```
#SET AX LINK [ <coupling_group>, <Slave> = <Master> {, <Slave> = <Master>} ]
```

or alternatively
\#AX LINK [NBR] [ <coupling_group>, <Slave> = <Master> \{, <Slave> = <Master>\} ]

```
<coupling_group>
<Slave>
<Master>
NBR
```

Number of the coupling group ${ }^{(1)}$
Designation or logical axis number of the slave spindle of coupling pair $\mathrm{i}^{(2)}$
Designation or logical axis number of the master spindle of coupling pair $i^{(2)}$
The logic switch NBR can change the evaluation from axis names to logical axes numbers. The axis couplings must then be defined with logical axis numbers.

In this case, the following rules apply in addition to Section Synchronous mode [> 355]:

- Coupling pairs are defined by the names or logical axis numbers of the spindles which are known in the channel. I.e. only spindles known in the channel can be linked.
- In the channel parameter list [1] [> 819]-2 coupling group 0 can be preset as the default group. This can be addressed directly with \#ENABLE AX LINK or \#AX LINK ON after start-up. It cannot be redefined in the NC program.


## Programing Example

## Synchronous spindle operation

Parameterisation in the channel parameter list [1] [> 819]: S (main spindle name S1), S1, S2, S3 The coupling pairs may be formed with spindle names S, S2 and S3.
Programming and selection/deselection of a spindle coupling:

```
N10 #SET AX LINK[1, S2=S, S3=S] (Main spindle is master for S2 and S3)
N20 #ENABLE AX LINK[1] (Select spindle couplings)
N30 S1000 M3 (Main spindle S+S2+S3 rotate cw 1000 rpm)
N40 #DISABLE AX LINK
or alternatively
N10 #AX LINK[1, S2=S, S3=S]
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF
or alternatively
N10 #AX LINK NBR[1, 11=6, 17=6] Coupling via log. axis numbers
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF
N50 M30 (End of program)
```

(1) 1 ... [Max. number of coupling groups-1], see [6] [> 819] -2.11
(2) Max. number of coupling pairs, see [6] [> 819]-2.12

- Different axis types may not be defined in a \#SET AX LINK or \#AX LINK command within a coupling pair. However, coupling pairs may form a coupling group with coupling pairs of another axis type.
Example:


## \#SET AX LINK [1, B=S, S2=X] WRONG

\#SET AX LINK [1, B=X, S2=S] PERMISSIBLE

- During an active coupling group, a spindle present in this group may not be declared the main spindle with \#MAIN SPINDLE [..], otherwise, this would result in inconsistencies between decoder and interpolator.
- Moreover, the programmer must be aware that using \#MAIN SPINDLE [..] may possibly mean that already defined coupling groups can no longer be activated since the spindle names are no longer consistent.
- The technology information for M03, M04, M05 is synchronised by the NC kernel both for the master spindle and for the slave spindle.
- External movement influencing of the master spindle with FEEDHOLD and OVERRIDE does not act on slave spindles. During active coupling the slave spindles also continue evaluating their own OVERRIDE and FEEDHOLD interfaces.
- Master and slave spindles move to the same absolute position with M19. The position may possibly not be reached simultaneously if the start position or the dynamic data of the spindles are different.
- Coupling is cancelled when the program is aborted.
- If a spindle is active as master or slave, it may only be commanded by the channel which activated the link.

Channel parameter list [1] [> 819]:

```
# Pre-assignment of possible axis links for synchronous mode
# ===========================================================
#synchro_data.koppel_gruppe[0].paar[0].log_achs_nr_slave 4
#synchro_data.koppel_gruppe[0].paar[0].log_achs_nr_master 1
#synchro_data.koppel_gruppe[0].paar[0].mode 0 ->AX_LINK
#synchro_data.koppel_gruppe[0].paar[1].log_achs_nr_slave 11
#synchro_data.koppel_gruppe[0].paar[1].log_achs_nr_master 6
#synchro_data.koppel_gruppe[0].paar[1].mode 1 ->SPDL_LINK
:
```


### 15.6 Cross-block synchronisation (Late Sync)

### 15.6.1 Implicit synchronisation

Spindle acceleration and deceleration operations may lead to substantial dead times in program execution since when the machine is at standstill, the spindle frequently needs to be set to the required speed first (e.g. M03 of type MVS_SVS) or during a positioning block with G00 (M3 of type MVS_SNS).

With M functions, implicit synchronisation provides the option of only checking acknowledgement if there is a switch over to a machining operation with G01/G02/G03/G151 etc. This reaction is achieved with synchronisation mode MVS_SLM. The identifier can only be used exclusively with other synchronisation (P-CHAN-00027).

## Programing Example

## Implicit synchronisation

:
N10 G00 M03 S1000 Z600
(M03: Synchronisation mode MVS_SLM)
N20 X100 Y100
N30 Z400
N40 G01 Z200
(Check whether M03 is acknowledged)
:

In N40, the interpolator checks for acknowledgement of the M function at the start of the braking instant. If the acknowledgement is output, there is no stop at block end. If no acknowledgement is output, deceleration occurs and if no acknowledgement is output by block end, the system stops at the target point.

It is possible to program further channel-specific M functions up to synchronisation by a motion block. Synchronisation of channel-specific M functions is handled entirely in parallel with axis-specific synchronisation.

### 15.6.2 Explicit synchronisation (\#EXPL SYN)

If G01 is used for positioning, no implicit synchronisation can be executed according to Sec. Implicit synchronisation [〉 689].
The \#EXPL SYN command is provided here for cross-block synchronisation and this permits explicit synchronisation of the $M$ function.

```
#EXPL SYN
```

(non-modal)

An M function which is to be synchronised with this additional command is defined with synchronisation mode MVS_SLP in the channel parameter list P-CHAN-00027. The identifier can only be used exclusively with other synchronisation (P-CHAN-00027.

## Programing Example

## Explicit synchronisation

:
N10 G01 M03 S1000 Z200 F5000 (M03: Synchronisation mode MVS_SLP)
N20 X100 Y100
N30 Z400
N40\# EXPL SYN (Check whether M03 is acknowledged)

At the braking instant, the path checks whether the acknowledgement has arrived based on the statement "\#EXPL SYN". A ramp-down occurs if this is not the case.

Further channel-specific and axis-specific M functions can be processed before the synchronisation command.

## $15.7 \quad$ Synchronisation of spindle M functions

Synchronisation between the interpolator and the relevant spindle is executed directly, i.e. the acknowledgement of M03, M04 (speed reached) and M05 (speed zero) is executed by the spindle itself. Bit PLC_INFO which can be set in addition to the existing synchronisation P-CHAN-00027 determines whether the PLC is also to be acknowledged. In this case, note the following:

In general the PLC automatically acknowledges each spindle M function for speed-controlled spindles. It is therefore not necessary to additionally set the PLC_INFO bit.
It is practical to use the PLC_INFO bit for position-controlled spindles. In this case, the PLC_INFO bit can also be set for each spindle $M$ function in addition to the synchronisation mode, thus causing the PLC to send an acknowledgement.

Channel parameter list [1] [ 819]:
Spindle S1 is to be a position-controlled spindle.

```
:
spindel[0].bezeichnung
spindel[0].log_achs_nr
spindel[0].s_synch
spindel[0].m\overline{3}_synch
spindel[0].m4_synch
spindel[0].m5_synch
spindel[0].m1\overline{9}_synch
spindel[0].s_prozess_zeit
spindel[0].m3_prozess_zeit
spindel[0].m4_prozess_zeit
spindel[0].m5_prozess_zeit
spindel[0].m1\overline{9}_prozes\overline{s}_zeit
:
```

S1
6


Fig. 167: Diagram of synchronisation of the spindle $M$ function

### 15.8 PLCopen programming

A complete list of PLCopen functions is contained in the overview of commands in the Appendix under PLCopen programming [> 692].
The scope of the Motion Control Platform (MCP) provides a number of function blocks (FB) for motion tasks. These FBs act on a single axis and are operated via the SPS. Each axis in the system is configured in the system as a so-called Single $\underline{\text { Axis }}$ Interpolator (SAI).

Alternatively these axes also can be addressed by the NC program because an SAI is always configured in the system as a conventional spindle. Special NC commands are therefore provided for the following FBs. These commands permit PLCopen-compliant programming in NC syntax.

MC_Home
MC_MoveAbsolute
MC_MoveAdditive
MC_MoveRelative
MC_MoveSuperImposed
MC_MoveVelocity
MC_Stop
MC_Gearln
MC_GearOut
MC_Phasing
MC_TouchProbe

## Homing

Axis motion to an absolute position
Relative axis motion to the commanded position
Relative axis motion to the current position
Relative axis motion to a motion already active
Endless axis motion at the specified velocity
Stop an axis motion
Gear coupling with a gear ratio
Release a gear coupling
Phase offset of couplings
Measurement of axis positions

The topology below displays the basic arrangement of SAI (spindle) axes within the overall system:


An SAI axis is addressed in the NC program in spindle-specific programming syntax. It must therefore be configured in the NC channel by its address letters and other data analogous to the configuration of a spindle in the channel parameter list. The most important settings in the channel parameter list are:

- spdl_anzahl (P-CHAN-00082) - Total number of (SAI) spindles
- bezeichnung (P-CHAN-00007) - Name of the (SAI) spindle
- log_achs_nr (P-CHAN-00036) - Logical axis number of the (SAI) spindle

For more information, see the documentation [1] [> 819] Section: Configuring spindles- and the Section Parametrising spindles [> 619].

## The PLCopen functions

- MC_MoveSuperlmposed
- MC_Gearln
- MC_GearOut
- MC_Phasing
- MC_TouchProbe
require additional specific SAI characteristics of the (spindle) axis which are configured in the axis parameters. The required settings are contained in the documentation [2] [ 819]-Section: SAI settings.

Each NC command of the corresponding FB is presented below. The syntax of these NC commands and the units of the programmed values are based on the corresponding input pins (VAR_INPUT) of the assigned FBs.

General syntax of an (SAI) NC command:

```
<spindle_name>[ <FB name> <Input_pin1> < Input_pin2> < Input_pin n...> {\ } ]
```

The axis name at the start of the NC command addresses the (SAI) spindle axis which is addressed by the NC channel.

The description of the input pins and the units and value ranges are also contained in the documentation [9] [> 819].

## Notice

All input pin values are programmed in metric units. In initial state the values must be specified in the specified internal units (e.g. $0.1 \mu \mathrm{~m}$ ). The parameter P-CHAN-00182 can be changed to specify values in default units (e.g. mm).

The input pin "Execute" is always assigned implicitly by programming the NC command. This is why no specific keyword is provided for this pin.
The line separator 'l' can be used within the [...] brackets to obtain a clear programming of the command over multiple lines.
The keywords "Id" and "WaitSyn" for job synchronisation in the NC program have no corresponding PINs in the PLC. The two keywords are available as of CNC Build V3.01.3100.01

By default PLCopen functions are executed irrespective of other NC program processing. There is no synchronisation between PLCopen single-axis jobs and path motion.
However, wait conditions can be defined to synchronise PLCopen functions with the program run. There are two options here:

1. Synchronisation at block end:

Defining the "WaitSyn" keyword causes the CNC to wait for the completion of the PLCopen job before continuing to the next NC block. If several PLCopen jobs are programmed in the NC line, continuation only takes place when all the jobs specified for the "WaitSyn" keyword are completed.

```
<spindle_name>[ <FB name> [WaitSyn] <input_pin2> <input pin n...> { \ } ]
```


## Programing Example

```
N10 G01 X100 F10000
N20 S[MC_MoveAbsolute WAIT_SYN POSITION=900000 ...]
;Continue to block N30 when spindle S has reached ;Position 90
N30 G01 X200 F100
```

1. Late synchronisation

The "Id" keyword can be used to assign a job ID to a PLCopen job. The \#WAIT MC_STATUS SYN [ID<JobNo>] command can wait for the end of the PLCopen job at a later time.

```
<spindle_name>[ <FB name> [ld<expr>] <input_pin2> <input pin n...> {\} ]
```

\#WAIT MC_STATUS SYN [ID<expr> \{ID<expr>\} ...]

If several PLCopen jobs with identical job numbers are started, the job number is assigned to the last PLCopen function commanded. It is then possible that the job last started may be subject to later synchronisation at job end.

## Programing Example

```
N10 G01 X100 F10000
N20 S[MC MoveAbsolute Id100 POSITION=900000 ...]
N30 G01 X}20
N40 S2[MC_MoveVelocity Id200 Velocity=10000 ...}
N50 G01 X300
N60 #WAIT MC_STATUS SYN [ID100 ID200]
; Continue to block N70 takes place when spindle S
; reaches position 90' and spindle S2 reaches the speed 10%/s
N70 G0 X0
```


### 15.8.1 MC_Home command

MC_Home commands a homing run for the axis. How an axis reacts to this command basically depends on the type of referencing operation.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

NC command:

```
<axis_name>[ MC_Home [ld=<expr>] [WaitSyn] {\}]
```


## Block diagram of the function block in PLCopen:



## Programing Example

MC_Home command

S [MC_Home]

### 15.8.2 MC_MoveAbsolute command

MC_MoveAbsolute commands an axis motion to an absolute position. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

## NC command:

```
<axis_name>[ MC_MoveAbsolute Position=<expr> Velocity=<expr> [Acceleration=<expr>]
    [Deceleration=<expr>] [Jerk=<expr>] Direction=<expr> [ld=<expr>] [WaitSyn] {\}]
```

Block diagram of the function block in PLCopen:

| Input pin | Unit | MC_MoveAbsolute |
| :---: | :---: | :---: |
| Position | [0.1 $\mu \mathrm{m}$ or $10^{-4}{ }^{\circ}$ ] | Axis $\qquad$ Axis |
| Velocity | [ $1 \mu \mathrm{~m} / \mathrm{s}$ or $10^{-3} / \mathrm{s}$ ] | - Position CommandAborted |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - Acceleration ErrorlD |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2]}$ | $\begin{aligned} & \text { - Deceleration } \\ & \text { Jerk } \end{aligned}$ |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] | Direction |
| Direction | 1 positive direction <br> 2 shortest path <br> 3 negative direction <br> 4 current direction |  |

Programing Example

S[MC_MoveAbsolute Position=133 Velocity=1000 Acceleration=500 \}
Deceleration=600 Jerk=20000 Direction=2]

### 15.8.3 MC_MoveAdditive command

MC_MoveAdditive commands a relative motion in addition to the commanded position if the axis is in "Discrete Motion" state. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
If the axis is in "Continuous Motion" state and receives a command from this command, the relative distance from the current position at the time of command is added.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

## NC command:

<axis_name>[ MC_MoveAdditive Distance=<expr> Velocity=<expr> [Acceleration=<expr>]
[Deceleration=<expr>] [Jerk=<expr>] [ld=<expr>] [WaitSyn] \{ <br>$] }$

Block diagram of the function block in PLCopen:

| Input pin | Unit | MC_MoveAdditive |
| :---: | :---: | :---: |
| Distance | [0.1 $\mu \mathrm{m}$ or $10^{-4}{ }^{\circ}$ ] | Axis - - - - - - - - - - Axis |
| Velocity | [ $1 \mu \mathrm{~m} / \mathrm{s}$ or $10^{-3} / \mathrm{s}$ ] | - Execute Done <br> - Distance CommandAborted |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - Velocity $\quad$ Error |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - Deceleration - Jerk |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] |  |

Programing Example
MC_MoveAdditive command

[^0]
### 15.8.4 MC_MoveRelative command

MC_MoveRelative commands a relative motion in addition to the current position. This is regardless of whether the axis is in "Discrete Motion" or "Continuous Motion" state. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

NC command:

```
<axis_name>[ MC_MoveRelative Distance=<expr> Velocity=<expr> [Acceleration=<expr>]
```

    [Deceleration=<expr>] [Jerk=<expr>] [Id=<expr>] [WaitSyn] \{1\}]
    
## Block diagram of the function block in PLCopen:

| Input pin | Unit | MC_MoveRelative |
| :---: | :---: | :---: |
| Distance | [0.1 $\mu \mathrm{m}$ or $10^{-4}{ }^{\circ}$ ] | Axis Axis |
| Velocity | [ $1 \mu \mathrm{~m} / \mathrm{s}$ or $10^{-3} / \mathrm{s}$ ] | - Distance CommandAborted |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - VelocityAcceleration Error |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | $\begin{aligned} & \text { - Deceleration } \\ & \text { Jerk } \end{aligned}$ |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] |  |

## Programing Example

MC_MoveRelative command

S[MC_MoveRelative Distance=321 Velocity=1200 Acceleration=555 \}
Deceleration=666 Jerk=22000]

### 15.8.5 MC_MoveSuperImposed command

MC_MoveSuperlmposed commands a relative motion in addition to a motion already active. The active motion is not interrupted but is superimposed over the commanded one. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
Since "Acceleration" values are also superimposed in the case of superimposed interpolation, corresponding axis parameters must be defined to ensure that the axis is not dynamically overloaded.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

## NC command:

<axis_name>[ MC_MoveSuperlmposed Distance=<expr> VelocityDiff=<expr> [Acceleration=<expr>] [Deceleration=<expr>] [Jerk=<expr>] [ld=<expr>] [WaitSyn] \{ <br>$] }$

## Block diagram of the function block in PLCopen:

| Input pin | Unit | MC_MoveSuperimposed |
| :---: | :---: | :---: |
| Distance | [0.1 $\mu \mathrm{m}$ or $10^{-4}{ }^{\circ}$ ] | Axis $----------\frac{\text { Axis }}{}$ |
| VelocityDiff | [ $1 \mu \mathrm{~m} / \mathrm{s}$ or $10^{-3} / \mathrm{s}$ ] | - Distance CommandAborted |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - VelocityDifion Error |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | $\begin{aligned} & \text { - Deceleration } \\ & \text { Jerk } \end{aligned}$ |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] |  |

## Programing Example

MC_MoveSuperImposed command

[^1]
### 15.8.6 MC_MoveVelocity command

MC_MoveVelocity commands an endless motion at the specified velocity. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
To stop the motion, the command must be interrupted by another command that sends a new command to the axis.

In conjunction with an MC_MoveSuperImposed command, the "InVelocity" output remains TRUE.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

## NC command:

<axis_name>[ MC_MoveVelocity Velocity=<expr> [Acceleration=<expr>] [Deceleration=<expr>] [Jerk=<expr>] Direction=<expr> [|d=<expr>] [WaitSyn] \{ <br>$] }$

## Block diagram of the function block in PLCopen:

| Input pin | Unit | MC MoveVelocity |
| :---: | :---: | :---: |
| Velocity | [ $1 \mu \mathrm{~m} / \mathrm{s}$ or $10^{-3} / \mathrm{s}$ ] | Axis - - - - - - - - - - |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - Velocity CommandAborted |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | - Acceleration Error |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] | - Jerk |
| Direction | 1 positive direction <br> 3 negative direction <br> 4 current direction |  |

Programing Example
MC_MoveVelocity command

```
S[MC MoveVelocity Velocity=1333 Acceleration=770 Deceleration=880 \
Jerk=10000 Direction=1]
```


### 15.8.7 MC_Stop command

MC_Stop leads to a controlled motion stop and places the axis in "Stopping" state. The motion stop is always jerk-limited at the constant set in "Jerk" to build up the deceleration rate.
If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
It aborts every ongoing command by other (SAI) motion commands.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

NC command:

```
<axis_name>[ MC_Stop [Deceleration=<expr>] [Jerk=<expr>] [Id=<expr>] [WaitSyn] {\}]
```

Block diagram of the function block in PLCopen:

| Input pin | Unit | MC_Stop |
| :---: | :---: | :---: |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | Axis - - - - - - - - - - Axis |
| Jerk | [1 mm/s ${ }^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] | - Deceleration Error <br> Jerk ErrorID |

## Programing Example

MC_Stop command

S[MC_Stop Deceleration=999 Jerk=25000]

### 15.8.8 MC_GearIn command

MC_Gearln commands a gear coupling at a gear ratio. The gear ratio defines the velocity ratio between master and slave axes. Synchronisation to the velocity is jerk-limited. "Jerk" is set in the command.

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.
The slave axis can be linked either to master setpoint values or to actual master values. The selection is made in the "Mode" parameter.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

The "PhaseShift" parameter specifies the required phase of master and slave axes which are set during automatic phase compensation (Mode $=256$ ). The value is programmed in metric units [ $0.1 \mu \mathrm{~m}$ or $10^{\wedge}-4^{\circ}$ ]. This parameter has the default value 0 .

## NC command:

<axis_name>[ MC_Gearln Master=<expr> RatioNumerator=<expr> RatioDenominator=<expr>
[Acceleration=<expr>] [Deceleration=<expr>] [Jerk=<expr>] Mode=<expr> [Id=<expr>] [WaitSyn] [PhaseShift=<expr>] \{ <br>$] }$

Block diagram of the function block in PLCopen:

| Input pin | Unit |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Master * | Logical axis number of the master axis |  |  |  |  |
| RatioNumerator * | Gear ratio numerator |  |  |  |  |
| RatioDenominator * | Gear ratio denominator |  |  |  |  |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] |  |  |  |  |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] |  |  |  |  |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] |  |  |  |  |
| Mode | 0 | Type of coupling between master axis and slave: <br> Coupling based on command values. |  |  |  |
|  | 128 | Type of coupling between master axis and slave: <br> Coupling based on actual values. |  |  |  |
|  | 256 | Automatic phase compensation: ON. |  |  |  |

*As a supplement to PLCopen the following options are available for these input pins:

| Master | As an alternative to the logical axis number the axis name of the <br> master spindle can also be programmed. |
| :--- | :--- |
| RatioNumerator | Alternative abbreviation RN |
| RatioDenominator | Alternative abbreviation RD |

## Programing Example

## MC_Gearln command

```
S[MC_GearIn Master=11 RatioNumerator=2 RatioDenominator=3 \
Acceleration=500 Deceleration=600 Jerk=20000 Mode=0]
Commanding with master axis names, default dynamic values and abbreviation of gear
ratio:
S[MC_GearIn Master=S2 RN=1 RD=3 PhaseShift=25 Mode=256 WaitSyn]
```


### 15.8.9 MC_GearOut command

MC_GearOut releases the coupling of the slave axis to the master axis that is specified by a velocity ratio. The current velocity of the slave is retained (endless motion).
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("ld" keyword) for later synchronisation.

## NC command:

```
<axis_name>[ MC_GearOut [Id=<expr>] [WaitSyn] {\}]
```

Block diagram of the function block in PLCopen:


## Programing Example

MC_GearOut command

S [MC_GearOut]

### 15.8.10 MC_Phasing command

MC_Phasing is used to achieve an offset of the slave axis relative to the master axis. Accordingly, a phase offset of the master axis is specified from the point of view of the slave axis and the slave axis tries to eliminate this offset by accelerating or decelerating. The motion is always executed jerk-limited at the constant set in "Jerk". This value is valid for both "Acceleration" and "Deceleration".

If the optional parameters "Acceleration", "Deceleration" and "Jerk" are not specified or set to $\leq 0$, the dynamic values are taken from the corresponding axis list.

The mechanical analogy is to release the coupling of the master and slave axes for a limited period.
When camming this command causes a change of the 'apparent' master position from the slave viewpoint. When gearing a phase shift between master and slave takes place by commanding a superimposed motion in the slave. Therefore with gearing, MC_Phasing has the same effect as MC_MoveSuperImposed (into which it is actually converted in the control system).

The dynamic values: "Velocity", "Acceleration" and "Deceleration" refer to the change in the "apparent" master position from the point of view of the slave with camming; but with gearing, they refer to the superimposed motion of the slave axis itself.
Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

## NC command:

<axis_name>[ MC_Phasing Master=<expr> PhaseShift=<expr> Velocity=<expr>
[Acceleration=<expr>] [Deceleration=<expr>] [Jerk=<expr>] [ld=<expr>] [WaitSyn] \{ <br>$] }$

## Block diagram of the function block in PLCopen:

| Input pin | Unit | MC_Phasing |
| :---: | :---: | :---: |
| Master | Logical axis number of the master axis | $\begin{array}{c\|c} \text { Master } \\ \text { Slave } \\ \text { Sla } \end{array}$ |
| PhaseShift | [0.1 $\mu \mathrm{m}$ or $10^{-4}{ }^{\circ}$ ] | - Execute CommandAborted |
| Velocity | [ $1 \mu \mathrm{~m} / \mathrm{s}$ or $10^{-3} / \mathrm{s}$ ] | - Velocity $r$ Error |
| Acceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] | $\begin{aligned} & \text { - Deceleration } \\ & \text { - Jerk } \end{aligned}$ |
| Deceleration | [ $1 \mathrm{~mm} / \mathrm{s}^{2}$ or $1^{\circ} / \mathrm{s}^{2}$ ] |  |
| Jerk | [ $1 \mathrm{~mm} / \mathrm{s}^{3}$ or $1^{\circ} / \mathrm{s}^{3}$ ] |  |

## Programing Example

## MC_Phasing command

[^2]
### 15.8.11 MC_TouchProbe command

The MC_TouchProbe command records an axis position when a trigger event occurs. The measurement channel and method (rising or falling edge of the trigger signal) are defined via the reference for the trigger signal source.

Optionally, the program can wait for the job to end ("WaitSyn" keyword) or can assign a Job ID ("Id" keyword) for later synchronisation.

## NC command:

<axis_name>[ MC_TouchProbe Channel=<expr> [Mode=<expr>] [ld=<expr>] [WaitSyn] \{ <br>$] }$

## Block diagram of the function block in PLCopen:

| Input pin |  | Unit |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| TriggerInput | Channel | ---- |  |  |  |
|  | Mode | --- | If no mode is specified, probe signal evaluated acc. to P -AXIS-00518 |  | - |
|  |  | 1 | Probe signal on falling edge |  |  |
|  |  | 2 | Probe signal on rising edge |  |  |

## Programing Example

## MC_TouchProbe

S[MC_TouchProbe Channel=1 Mode=2]

## 16 Macroprogramming (\# INIT MACRO TAB)

Macros permit the assignment of an alias name (macro name) to an executable NC code (macro content). The macro content may consist of arithmetic expressions and NC commands. The related NC code is executed whenever the corresponding macro name (macro call) is specified.

Macros support the generation of low-maintenance and clear NC programs. Changes need only be made once in the macro content.

Macros may only be configured in the channel parameter list [1] [> 819]-1 and also defined in the NC program.

## Syntax of a macro definition

```
"<macro name>" = "<macro content>"
```

<macro name> name of the macro (alias)
<macro content> executable NC code

- Macro name and macro content must be enclosed in inverted commas "...".
- The macro name makes a distinction between uppercase and lowercase letters.
- The redefinition of predefined macros is configurable [6] [ 819]-6.39.
- Macro definition remain valid after program end (M30) or CNC reset. They are usable until the next controller start-up or \#INIT MACRO TAB.
- Macro definitions that were configured in the channel parameter list cannot be deleted by \#INIT MACRO TAB.
- The maximum number of macro definitions [6] [> 819]-6.25 and their length [6] [ 7 819]-6.37/-6.38 as fixed. The values are settable as of Builds V3.1.3079.17 and V3.1.3107.10:
- Maximum number of macros P-CHAN-00509
- Maximum number of predefined macros P-CHAN-00510
- Maximum number of characters in macro name P-CHAN-00511
- Maximum number of characters in macro content P-CHAN-00512
- Macro definitions and other NC commands can be combined in the same NC block.

A macro call is initiated by the macro name and can be combined in the same NC block with other NC commands.

## Syntax of a macro call:

```
"<macro name>"
```


## Programing Example

macros

```
N10 "POSITION 1" = "X200 Y200 Z300" (macro definition)
N20 "POSITION`2" = "X300 Y100 Z50" (macro definition)
N200 "POSITION_1" (macro call, X200 Y200 Z300 is executed)
:
```

```
N500 "POSITION_2" (macro call, X300 Y100 Z50 is executed)
```


## Release Note

As of Build V2.11.2010.02 replaces the command \#INIT MACRO TAB the command \#INIT MAKRO TAB. For compatibility reasons, this command is still available but it recommended not to use it in new NC programs.

The following command must be used

## \#INIT MACRO TAB

as an alternative. This command clears all macros previously defined in the macro table. The macros pre-assigned by the channel parameters [1] [> 819]-1 are retained.

### 16.1 Nesting macros

The use of macros on the right-hand side of the instruction (so-called Nesting) is permitted in combination with NC code. The maximum nesting depth is fixed [6] [ 819]-6.40.

Nesting is displayed by a 'l' character that precedes the delimiting quotation marks. A macro should always represent complete expressions of an NC block (NC command, mathematical expression, term). This excludes the possibility of a macro only represented by the address letter of an NC command without the related mathematical expression. This connection will be discussed in the next sections.
"<Macro_name>" = "<NC_Code> \"<Macro_name_i> \" <NC_Code>"

## Attention

A macro may not be called by its own nested macro name. Only nesting of other macro names is permitted.

## Programing Example

## Nesting macros

```
Example 1:
N10 "POS 1" = "X500 Y200" (Macro definition)
N20 "MOVE1" = "GO1 \"POS 1\" F1000" (Macro definition with nesting)
N30 "MOVE1'
Macro call)
M30
Example 2:
N10 " STRING_1 " = " 5*12 " (Macro definitions)
N20 " STRING 2 " = " G \"STRING 1\" + 5 "
N30 " STRING_3 " = " M \" STRING_1\ " \" STRING_2 \" "
:
N200 " STRING_3" (Call the nested macro)
(Corresponds to: N200 M60 G65
```


### 16.2 Use in mathematical expressions

Macro names may be assigned to arithmetic expressions and parts of them. Recursive treatment (nesting) may also be used within mathematical expressions.

It must be ensured that a string in a macro content always combines complete levels (i.e. terms whose results are not influenced by inserting '[' at the start and ']' at the end).

## Programing Example

Use in mathematical expressions

```
Right:
N10 "STRING1" = "0.5"
N20 "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"
N40 X[-2 * "STRING1" + "STRING2" + "STRING3"] (move to X60)
M30
Wrong: Macros contain only incomplete mathematical expressions
N10 "STRING1" = "COS["
N20 "STRING2" = "90]"
N30 "STRING3" = " \"STRING1\" \"STRING2\" " Error
```


### 16.3 Separating address letter and mathematical expression

It is also possible to only assign a macro name the address letter and program or define the mathematical expression in a second macro.
This permits macros that reference the main axes to be defined in a higher-level NC program. The macro may then be used in the subroutine or cycle, providing a certain independence from the selected processing level.

Programing Example
Separating address letter and mathematical expression

Only the address letter is contained in the macro content. The mathematical expression is programmed or defined in a separate macro:
"1.PA" = "X" "2.PA" = "Y" primary axis)
"Ziel 1.HA" = "V.E.POS1 + P12"
"1.PA" "TARGETl_1.PA" "2.PA" 100

### 16.4 Restrictions

No end-of-line or end-of-string ( ${ }^{\prime} 10^{\prime}$ ) characters may be included in macro content. The limits the macro definition to one block.

```
"Macro_Move" = "X100 G01 \0"
"Macro_Move2" = "X100
        G01"
```

...
M30

Macro content may not contain any control block statements (\$).

```
"IF" = "$IF"
"END_IF" = "$ENDIF"
P1 = 0
"IF" P1 == 0
P2 = 2
"ENDIF"
M30
```

Macro content may not contain any strong constants. However, strong functions or V.E. variables of the string type are permitted.

Recursive macro calls cause an error when the macro is executed.

```
"Macro_Recursive" = "G01 X100 \"Macro_Recursive\""
M30
```


## 17 5-Axis functionality

### 17.1 Rotation Tool Centre Point (RTCP)(\# TRAFO OFF)

## Notice

The use of this feature requires a license for the "Transformations" extension package. It is not included in the scope of the standard license.

| \#TRAFO ON | Select RTCP | (modal) |
| :--- | :--- | :---: |
| \#TRAFO OFF | Deselect RTCP | (modal) |

The RTCP function represents tool compensation in space.
After RTCP is selected, the contact point of the current tool remains stationary relative to the workpiece when tool orientation is changed (please note the assignments of the tool tip offset parameters P-CHAN-00094 and P-TOOL-00009 when assigning parameter values to kinematic transformation).


Fig. 168: Motion control with/without RTCP

The resulting motion is shown on the left of the figure when the rotary machine axis is moved.
RTCP shifts the centre point of rotation to the tool tip (centre point of tool rotation). The resulting offsets in the axes $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ based on tool motion are compensated in each cycle by corresponding opposing motions.

Only the axes $\mathrm{X}, \mathrm{Y}, \mathrm{Z}$ are outputs from kinematic transformation; the rotary machine axes are programmed as usual.

The RTCP function may not be selected when TLC is active (see Section Tool Length Compensation (\#TLC ON/OFF) [ 715]).

## Attention

When kinematic transformation is active, axis-specific tool offsets in ax_versatz[<ax_index>] (P-TOOL-00006) are only taken into consideration if axes are not influenced by the transformation function. Depending on the transformation type, they typically refer to all axes with index $>2$ when RTCP is used.

The axis-specific tool offsets of the first three axes (index 0, 1, 2) are not taken into account when transformation is active. If tool offsets should also be effective for these axes when transformation is active, enter the values in the kinematic offsets of the tool (P-TOOL-00009) mentioned above.

## Programing Example

## RTCP example 1

```
N10 T1 D1
(Select 2.5D tool compensation)
N20 #TRAFO ON
(Select RTCP)
(Programming rotary axes modifies tool
(Orientation. Tool contact point remains)
(stationary)
N40...
.
N100 #TRAFO OFF (Deselect RTCP)
N200 M30
```


## Programing Example

RTCP example 2

```
N10 #KIN ID [1]
(Select machine kinematic)
N20 #TRAFO ON
N30 G01 G18 X20 Y0 Z25 B90 F500 (Move tool to right)
(Select RTCP)
N40 G91 X-8 (Move to start position)
N50 G90 G02 X-12 I-12 B-90 F2000 (Process)
N60 ...
```



Fig. 169: Motion control with RTCP

### 17.2 Tool Length Compensation (\#TLC ON/OFF)

## Notice

Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

```
#TLC ON [ <delta_tool_length> ]
#TLC OFF
\begin{tabular}{lr} 
Select TLC & (modal) \\
Deselect TLC & (modal)
\end{tabular}
```

<delta_tool_length> Tool length difference in [mm, inch].

TLC permits the reuse of NC programs which were created by a programming system and which consider a specific tool length, even if the tool length was changed on the machine. It must be noted that no new tool offsets or radii may be compensated.

## Notice

The TLC function may not be selected when RTCP is active.
The error is output with ID 20669. The two functions mutually exclude each other.


Fig. 170: When the tool length is changed, TLC transforms $\Delta L$ in each cycle.

## Programing Example

## Tool length compensation)

```
N10 #TLC ON [15] (Select TLC with extended tool, \triangleTL =15 mm))
N20 .
N100 #TLC OFF (Deselect TLC)
N200 #TLC ON [-20] (Select TLC with extended tool, \triangleTL =-20 mm))
N210 .
N300 #TLC OFF (Deselect TLC)
N200 M30
```


### 17.3 Orienting tool (\#TOOL ORI CS)

The first motion block that follows \#TOOL ORI CS aligns the tool in parallel to the 3rd main axis of current PCS (also $\mathrm{W}_{0}$ - or MCS possible). Programming the rotary axes in this block overrides the positions that are valid for orientation.


Fig. 171: Tool aligned perpendicularly to the $X-Y$ machining plane

## Programing Example

Align tool

```
N10 B10 C20
    (Slant tool)
N20 #TOOL ORI CS (in next motion block, align tool in parallel to)
    (the Z axis of current PCS, here MCS)
    (Motion block in the MKS, tool is aligned B=O,C=O)
    (Slant tool)
    (Align tool in next motion block)
    (Select RTCP)
N70 #CS ON[0,0,0,-80,-30,45] (Transition to a rotated PCS)
N80 X100 (Motion block in PCS). align tool from N50, refers)
    (to the MCS, i.e. B=0,C=0)
N90 #TOOL ORI CS (Align tool in next motion block)
N100 Y150 (Too aligned in parallel to Z axis of PCS)
```

N110 \#TOOL ORI CS
N120 Z100 B45 C10 (\#TOOL ORI CS has no effect when programming) (rotary axes)
N130 G18 (Change to the Z-X interpolation plane)
N140 \#TOOL ORI CS (Align tool in parallel to the Y axis)
N150 X0
(Tool aligned perpendicularly to $X-Z$ machining plane)
M30

### 17.4 Machine kinematics (\#KIN ID)

## Notice

Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

The following statement is used to define the machine/tool head kinematics:
\#KIN ID [ [<expr>] ] ] Definition of machine/tool head kinematics (modal)
<expr> Kinematic ID The purpose of the kinematic ID is to identify the machine or tool head-specific kinematic types implemented in the controller. Their default setting after controller start-up is parameterised in P -CHAN-00032.

The default kinematic ID is set by programming \#KIN ID without parameters.
In addition, a kinematic change can be executed automatically when the tool is changed by assigning the "kin_id" element in the tool data list.

An unknown kinematic ID causes the output of an error message and a decoding stop when RTCP, TLC or TOOL ORI CS is selected.

When kinematic ID 0 is selected, no kinematic is enabled with no warning or error message.

## Attention

It is not permitted to change kinematics with \#KIN ID... when RTCP or TLC is active.

## Programing Example

Machine kinematics

```
N10 #TOOL ORI CS (Align tool, default kinematic)
(valid from P-CHAN-00032.)
N20 T1 D1 (Select tool)
N30 #TRAFO ON (Select RTCP, default kinematic)
(valid from P-CHAN-00032)
N40 #TRAFO ON (Deselect RTCP)
N50 #KIN ID[2] (Select kinematic which is saved in the)
(internal library under ID '2' )
N60 #TRAFO ON (Select RTCP, kinematic 2)
N70 #TRAFO ON (Deselect RTCP)
N80 M30
```


### 17.5 Modify kinematic characteristics (\#KIN DATA)

This command permits characteristics of the active kinematics to be modified when the transformation is active (\#TRAFO ON [ $>713]$ ). This includes, for example, the possibility of specifying which axes should not move in cases of redundant degrees of freedom of a kinematic system.

## Release Note

This function block is available as of CNC Build V3.1.3080.

## Notice

Only kinematic ID 210 is supported.
If the command is used with other kinematics, it has no effect.

```
#KIN DATA [LOCKDOF { AX=..}| { AXNR=..} ]
(Select axes to be locked)
#KIN DATA [UNLOCKDOF { AX=..}|{ AXNR=..}] (Select axes to be enabled)
```

| LOCKDOF | Set the entry in parameter P-CHAN-00458 for the specified axes. As opposed to the vari- <br> able V.G.KIN[i].LOCK_DOF[<AXIDX>], the command can also be used with active trans- <br> formation. |
| :--- | :--- |
| UNLOCKDOF | Reset the entry in parameter P-CHAN-00458 for the specified axes. As opposed to the <br> variable V.G.KIN[i].LOCK_DOF[<AXIDX>], the command can also be used with active <br> transformation. |
| AX=<axis name> | Names of axes to be locked/enabled <br> AXNR=<expr> |

## Example

</> Using the \#KIN DATA command

The starting point for the following programming example is a simplified configuration of a coupling kinematic with a stationary robot on an $X$ linear axis. When TCP is programmed (axis identifier X), the X linear axis moves first due to the motion priority in P-CHAN-00450.

Extract from parameterisation of channel parameters:

```
gruppe[0].achse[00].log_achs_nr 1
gruppe[0].achse[00].bezeichnüng X
gruppe[0].achse[00].default_feed_axis 0
gruppe[0].achse[06].log achs nr 7
gruppe[0].achse[06].bezeichnung X_LIN
gruppe[0].achse[06].default_feed_axis 0
gruppe[0].achse[07].log_achs_nr \overline{8}
```

```
gruppe[0].achse[07].bezeichnung X_ROB
```

```
gruppe[0].achse[07].default_feed_axis 0
```

```
gruppe[0].achse[07].default_feed_axis 0
```

...
trafo[0].id 210
trafo[0].group[0].name LIN_ROB
trafo[0].group [0].chain[0] LIN
trafo[0].group [0].chain[1] ROB
trafo[0].group [0].move_prio[0] LIN
trafo[0].group [0].move_prio[1] ROB
trafo[1].id 91
trafo[1].name LIN
trafo[2].id 45
trafo[2]. name ROBOT

The example below locks the X linear axis in block N01. The robot can then only move the programmed motion.

The X linear axis is unlocked in block N03. The axis then moves the program X motion since it has a higher motion priority than the robot.

```
NO1 #KIN DATA [LOCKDOF AXNR=7]
N02 G00 G90 X1500
( Cartesian axis positions: X=1500, X_LIN=0, X_ROB=0
NO3 #KIN DATA [UNLOCKDOF AX=X_LIN]
NO2 G00 G90 X1000
( Cartesian axis positions: X=1000, X_LIN=-500, X_ROB=1500
```


### 17.6 Positioning without compensation motion (\#PTP ON/OFF, \#AX LOCK ALL, \#AX UNLOCK ALL)

Notice
Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

| \#PTP ON | Select transformation PTP motion control | (modal) |
| :--- | :--- | :--- |
| \#PTP OFF | Deselect transformation PTP motion control |  |

Tool positioning and alignment after kinematic transformation is selected causes a compensation motion in the machine axes because the tool centre point (TCP) is moved on the path. If these compensation motions are undesired, a more time-optimised motion can be executed with the commands listed above.

For 5-axis machines the motion is based on the reference point of the tool head; - in contrast to the TCP, the reference point moves on a straight line at the programmed feedrate ( F word) or at rapid traverse (see Fig. blow).

On non-Cartesian machine structures (e.g. robots, tripods) neither the TCP nor the reference point moves on a straight line. The programmed feed (F word) or rapid traverse acts on the machine axes. However at the end of the motion, it is ensured that the TCP is located on the programmed PCS target point.

Motion programming is identical to PCS programming; the controller executes the conversion of PCS coordinates into MCS coordinates. As opposed to using the \#WCS TO MCS command, adopting offsets and tool data is executed in the same way as with active kinematic transformation.

## Attention

When PTP motion control is active, ACS values are displayed in the PCS coordinates in the realtime part of CNC.


Fig. 172: Motion control with/without \#PTP

## Attention

When PTP motion control is active, it is not permitted to select or deselect additional coordinate transformations (\#(A)CS ON/OFF, \#MCS ON/OFF etc.).

## Programing Example

## Positioning without compensation motion (PTP)

## Example of previous figure;:

```
N10 T1 D1 ;Select tool
N20 G00 X-200 Y0 Z0 A90 ;MCS start position
N30 #TRAFO ON ; Select RTCP, default kinematic
;valid from P-CHAN-00032
;Transformation PTP motion on
N50 G00 X200 Y0 Z0 A-90 ; PCS target point
N180 #PTP OFF ;Transformation PTP motion off
N185 G01 X100 Y150 F5000
N500 #TRAFO OFF ; Deselect RTCP
N999 M30
```

Example with automatic tool alignment:

```
N10 T1 D1
;Select tool
N20 #TRAFO ON ;Select RTCP, default kinematic)
;valid from P-CHAN-00032
N30 #TOOL ORI CS ;Align tool, default kinematic
;valid from P-CHAN-00032.
N40 #PTP ON ;Transformation PTP motion on
N50 G00 X0 Y0 Z100 G90 ;PCS start position
N180 #PTP OFF
;Transformation PTP motion Off
N185 G01 X100 Y150 F5000
N500 #TRAFO OFF
;Deselect RTCP
N999 M30
```


## Notice

In conjunction with PTP motion control, it may sometimes be necessary to suppress the resulting path motions for certain axes for technological reasons (e.g. In order to remain within the limits of software limit switches).
For this purpose, the \#AX LOCK/UNLOCK ALL command temporarily locks motions of single axes when PTP programming is active.

```
#AX LOCK [ { AX<Achsname> } | { AXNR<expr> } ] Select axes for locking

AX<Achsname>
AXNR<expr>
ALL

Name of axes to be locked
Logical numbers of axes to be locked, positive integers
Release locked axes. If several axes are locked, they can only be released together.
The release of only one specific axis is not possible.
If only one single axis is locked, it must also be released with ALL.

Programming of \#AX LOCK, \#AX UNLOCK ALL is only permitted when \#PTP is active.
Locked ACS output axes of the kinematic transformation do not move with G00 and G01.

\section*{Programing Example}

\section*{Positioning without compensation motion (PTP)}

Select the axis to be locked or release all locked axes.
```

Correct:
N10 \#CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N20 \#PTP ON
N30 \#AX LOCK[AX=Z AX=B] ;alternative: \#AX LOCK[AXNR=3 AXNR=5]
N40 G00 G90 U30
N60 \#AX UNLOCK ALL ;With implicit position request
N70 \#PTP OFF
N80 G01 G90 Z0 F3000
N90 G01 U40 F2000

```

\section*{Wrong:}
```

N10 \#AX LOCK[AX=Z] ; Programming before PTP
N20 \#CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N30 \#PTP ON
N40 G00 G90 U30
N60 \#AX UNLOCK ALL ;With implicit position request
N70 \#PTP OFF
N80 G01
N200 \#CYL OFF

```

\subsection*{17.7 Coordinate systems}

The number of possible coordinate systems is limited to [SYSP// Number 6.17].

\subsection*{17.7.1 Defining a machining coordinate system (\#CS DEF, \#CS ON/OFF, \#CS MODE ON/OFF)}

Defining and storing a CS:
\#CS DEF [ [<CS-ID>] ] [ <v1>, <v2>, <v3>, < 41\(\rangle,\langle\varphi 2\rangle,<\varphi 3\rangle\) ]

Define and store with simultaneous activation:
\#CS ON [ [<CS-ID>] ] [ <v1>, <v2>, <v3>, < \(1>,<\varphi 2>,<\varphi 3>\) ]
(modal)
\#CS ON [<CS-ID>] \(\quad\) Select a saved CS (modal)
\#CS ON Select last CS defined (modal)
\#CS OFF Select last CS activated (modal)

The CS-ID parameter may not be programmed here since it is only permitted to deselect the last CS activated.
<CS-ID> Coordinate system ID. The CS-ID is assigned the default value 1 at program start. If the CS-ID is not programmed with \#CS DEF or \#CS ON, the next free CS-ID is calculated automatically. However, a CS of this type is not longer available after it is deselected with \#CS OFF! At the same time, a maximum of 5 CS definitions can be stored.
<vi> Components of the translatory offset vector in [mm, inch]. (These refer to the main axes in the sequence contained in G17).
\(<\varphi i>\quad\) Angle of rotation in [ \({ }^{\circ}\) ].
\#CS OFF ALL Deselect all machining coordinate systems CS

A CS (PCS processing coordinate system) is characterised by the relative offset (V2 in figure below) and the rotation relative to the current work piece coordinate system (WCS). Current zero offset, clamp position offset and reference point offset (V1 in figure below) determine the position of the CS relative to the machine coordinate system (MCS).


Fig. 173: Machining on an inclined plane

\section*{Default setting of rotation sequence and rotation mode:}

If the rotations \(\varphi 1, \varphi 2\) and \(\varphi 3\) are defined, they are executed in the default setting in the mathematical positive direction (figure below) in the sequence as listed below:
1st rotation at angle \(\varphi 3\) about the 3rd axis (e.g. z)
2nd rotation at angle \(\varphi 2\) about the new 2 nd axis (e.g. y')
3rd rotation at angle \(\varphi 1\) about the 1 st axis (e.g. \(\mathrm{x}^{\prime \prime}\) )

This rotation sequence is also referred to as YAW - PITCH - ROLL. The rotations always refer here to the new axes of the currently rotated CS (rotation mode).
(The specified axis sequence of the axes always corresponds to the sequence of the main axes at G17, irrespective of G17/G18/G19).


Fig. 174: Definition of a CS by 3 rotations referred to the new axes

\section*{Free definition of rotation sequence and rotation mode:}

Every orientation in space can be reached by concatenating three basic rotations. There are 6 possible rotation sequences about 2 axes (known as classic Euler angles) and 6 rotation sequences about 3 axes (known as Tait-Bryan angles).
Rotations either refer to fixed axes in space (extrinsic rotation) or to the new axes of the currently rotated CS (intrinsic rotation). The figure below shows this difference compared to the figure above.


Fig. 175: Definition of a CS by 3 rotations about fixed axes in space

The rotation sequence can be configured with P-CHAN-00394. This rotation sequence can be changed in the NC program by the following command:
```

\#CS MODE ON [ROTATION_SEQUENCE=<rot_sequence>]
(modal)

```
<rot_sequence>
Rotation sequence as string according to:

Euler angles
XYX, XZX, YXY, YZY, ZXZ, ZYZ

Tait-Bryan angles
XYZ, XZY, YXZ, YZX, ZXY, ZYX (Default)

Deselecting and restoring default setting by:
```

\#CS MODE OFF [ROTATION_SEQUENCE]
(modal)

```

In the default setting, rotation is always executed about the new axis of the currently rotated CS (YAW - PITCH - ROLL). The following command can change this rotation mode in the NC program to rotations about fixed axes in space:
\#CS MODE ON [ROTATION_MODE_FIXED] (modal)

ROTATION_MODE_FIXED
Rotation about (fixed) axes in space of the coordinate system at rotation state

Deselecting and restoring default setting by:
```

\#CS MODE OFF [ROTATION_MODE_FIXED]

Without \#CS MODE OFF [...] all settings remain active until main program end (M30) or RESET. After next program start, the default settings are again valid.

The position (with respect to the current WKS) of a CS that has been defined via \#CS DEF [<CSID>] [...] or CS ON [<CS-ID>] [...] is stored and can be re-selected via \#CS ON without a specification of parameters. However, if the overall offset in the MCS is modified in the meantime, the CS has a new position relative to the MCS.

Zero offsets and reference point offsets may be programmed in the CS during machining. However, these values are only valid until the CS is deselected and are not saved.

The axis designations are retained in the CS.

## Programing Example

## Example 1

```
N005 P1 = 2
N010 #CS DEF [1][P1,15,5,20,30,45] (Define and store a CS)
    (with ID 1)
    (Relative offsets: X2, Y15, Z5)
    (Rotations: 45* about Z,30* about Y', 20* about
X'')
NO20 #CS ON[1]
:
:
N100 #CS OFF
:
:
N200 P1=10
N210 #CS ON [P1,15,5,2,3,60]
    (Define and activate a CS)
    (with the automatically defined ID 2)
:
:
N300 #CS OFF (Deselect the last CS activated (ID 2))
    (Then the CS is deleted with ID2.)
:
N400 M30
```


## Programing Example

Example 2

```
N05 P1 = 2
N10 #CS DEF [3][P1,15,5,2,3,4.5]
N20 #CS DEF [2][P1,15,5,2,3,4.5]
(Define and store)
(a CS with ID 3)
(Define and store)
(a CS with ID 2)
```

```
N30 #CS DEF [5][0,1,2,0,30,30
(a CS with ID 5)
N30 #CS ON
    (Activate the CS with the)
    (last ID 5 programmed)
:
N50 #CS OFF
N60 #CS ON[3] (Activate CS with ID 3)
:
N80 #CS OFF
N90 #CS DEF [3][1.1. 2,1,3,0,0.33] (Redefine the CS with ID 3)
:
M30
```


## Programing Example

## Example 3

If several coordinate systems are selected in succession, e.g. with CS ON [...] (without CS_ID), they form a new linked total CS. This must be deselected step by step by a corresponding \#CS OFF.

It is permitted to select combined CS's with and without CS IDs but this is not recommended for the sake of program clarity.

Example of multiple programming of CS (without CS_ID):

```
NO10 #CS ON [0,0,0,0,0,20] (Define and activate a CS with)
    (automatically defined ID 1)
    (No offsets, only rotation 20* about Z)
:
N050 #CS ON [0,0,0,0,0,30] (Define and activate a CS with)
(automatically defined ID 2)
(No offsets, only rotation 30* about Z)
->(This results in a total CS with a rotation of 50* about Z)
:
N100 #CS OFF (Deselect the CS with ID 2, then the CS is)
    (deleted with ID 2.)
->(CS with ID 1 with a rotation of 20' about Z remains active)
:
:
N200 #CS OFF (Deselect the CS with ID 1, then the CS is)
    (is deleted with ID 1 and all CS's are deselected.)
:
:
N400 M30
```


### 17.7.2 Defining/activating a coordinate system for fixture adaptation (\#ACS)

The fixture adaptation coordinate system (ACS) compensates a sloping position of the workpiece or workpiece palette. It is defined, selected and deselected in the same way as the machining coordinate system (CS).

Defining and storing an ACS:
\#ACS DEF [ [<ACS-ID>] ] [ <v1>, <v2>, <v3>, < $1>,<\varphi 2>,<\varphi 3>$ ] (modal)
Define and store with simultaneous activation:
\#ACS ON [ [<ACS-ID>] ] [ <v1>, <v2>, <v3>, < $\varphi 1>,<\varphi 2>,\langle\varphi 3>$ ]
(modal)
\#ACS ON $[<A C S-I D>] \quad$ Select a stored ACS (modal)
\#ACS ON $\quad$ Select last ACS defined (modal)

## \#ACS OFF <br> Deselect last activated ACS <br> (modal)

Parameter ACS-ID may not be programmed here since it is only permitted to deselect the last ACS activated.
<ACS-ID> Coordinate system ID. The ACS-ID is assigned the default value 1 at program start. If the CS-ID is not programmed with \#CS DEF or \#CS ON, the next free CS-ID is calculated automatically. ACS programmed in this way are not available again after they are deselected with \#ACS OFF.
<vi> Components of the translatory offset vector in [mm, inch]. (These refer to the main axes in the sequence contained in G17).
$<\varphi i>\quad$ Angle of rotation in [ ${ }^{\circ}$ ].

The ACS is modal and may be selected and deselected independent of a CS.
Zero offsets and reference point offsets may be programmed in the ACS. However, they are only valid until the ACS is deselected and they are not stored.

## Programing Example



```
N005 P1 = 2
NO10 #ACS DEF [1][P1,15,5,20,30,45](Define and store an ACS)
                                    (with ID 1:-)
                                    (Relative offsets: X2, Y15, Z5)
                                    (Rotations: 45 about Z, 30' about Y', 20' about
X'')
NO20 #ACS ON[1] (Activate ACS with ID 1)
:
:
N100 #ACS OFF (Deselect ACS with ID 1)
:
N200 P1=10
N210 #ACS ON [P1,15,5,2,3,60] (Define and activate an ACS)
    (with the automatically defined ID 2)
:
:
N300 #ACS OFF (Deselect the last ACS activated (ID2))
(Then the ACS is deleted with ID 2)
:
N400 M30
```


## Programing Example

## ACS example 2

```
N5 P1 = 2
N10 #ACS DEF [1][10,15,5,2,3,4.5] (Define and store)
    (an ACS with ID 1)
    (Define and store)
    (an ACS with ID 3)
N30 #ACS DEF [P1+3][2*P1,1,2,0,30,30] (Define and store)
(an ACS with ID 5)
N30 #ACS ON
    (Activate the ACS with the)
    (last ID 5 programmed)
N50 #ACS OFF
N60 #ACS ON[3] (Activate ACS with ID 3)
N80 #ACS OFF
N90 #ACS DEF [3][0.1. 2,1,3,0,0.3] (Redefine the ACS with ID 3)
M30
```


## Programing Example

## ACS example 3

If several coordinate systems are selected in succession, e.g. with ACS ON [...] (without ACS_ID), they form a new linked total ACS. This must be deselected step by step by corresponding \#ACS OFF.

It is permitted to select the combined ACS with and without ACS IDs but it is not recommended for the sake of NC program clarity.

Example of multiple programming of ACS (without ACS_ID):

```
N010 #ACS ON [0,0,0,0,0,20] (Define and activate an ACS with)
    (automatically defined ID 1)
    (No offsets, only rotation 20* about Z)
NO2O #ACS ON [0,0,0,0,0,30] (Define and activate an ACS with)
    (automatically defined ID 2)
    (No offsets, only rotation 30' about Z)
```

->(This results in an ACS with a total rotation of $50^{\circ}$ about Z )
:
N100 \#ACS OFF (Deselect the ACS with ID 2, then the ACS is)
(deleted with ID 2.)
->(ACS with ID 1 with a rotation of $20^{\circ}$ about $Z$ remains active)
:
:
N200 \#ACS OFF (Deselect the ACS with ID 1, then the ACS is)
(is deleted with ID 1 and all ACS are deselected.)
:
N400 M30

### 17.7.3 Linkage of coordinate systems

New coordinate transformations may be formed by combining ACS and CS.


Fig. 176: The combination of ACS and CS permits machining on an inclined plane with a slanted clamped workpiece.

Several ACS and CS are linked separately in the sequence they are selected. The resulting ACS is then linked to the resulting CS for overall transformation. Linkage always takes place with the ACS first irrespective of programming.

A maximum of 10 ACS/CS combinations can be linked to form an overall transformation.

Individual ACS's are deselected in the opposite sequence to selection. The same applies to the CS. To simplify this, \#(A)CS OFF does not use an ID parameter for programming (see the two Figs in Defining a machining coordinate system (\#CS DEF, \#CS ON/OFF, \#CS MODE ON/OFF) [ $\left.{ }^{-} 727\right]$ ).

## Programing Example

## Linkage of coordinate systems

```
N100 #CS ON [1]
N110 #ACS ON [2]
N120 #ACS ON [1]
N130 #CS ON [2]
N140 #ACS OFF
N140 #CS OFF
N150 #ACS OFF
N160 #CS OFF
M30
```

```
( CS[1])
(ACS[2] O CS[1])
(ACS[2] ○ ACS[1] ○ CS[1])
(ACS[2] ○ ACS[1] ○ CS[1] O CS[2])
(ACS[2] ○ CS[1] O CS[2])
(ACS[2] O CS[1])
( CS[1])
```



Fig. 177: Activating or changing the ACS without deselecting the CS's which are already active

It must be noted that the relative linkage of ACS or CS may in general result in a changed sequence of selection and lead to different results (see figure below).


Fig. 178: Result of a CS linkage depending on the sequence of selection (CS[1]-CS[2] or CS[2] - CS[1]).

The CS (or ACS) with the same ID may also be selected several times and linked to itself.

## Programing Example

Linkage of coordinate systems

```
N10 #CS DEF[1][0,0,0,0,0,20]
N20 LL TEILEPRG (Contour in the system X-Y)
N30 #CS ON[1]
N40 LL TEILEPRG
N50 #CS ON[1]
N60 LL TEILEPRG
N70 #CS OFF
```



Fig. 179: Linkage of coordinate systems

The following NC commands store the currently active overall transformation:

```
#CS DEF ACT [<CS_ID>]
#ACS DEF ACT [<ACS_ID>]
```

As opposed to sequential deselection of the (A)CS by (A)CS OFF, the following NC commands can directly deselect the partial transformations formed from the linkage of CS or ACS.

```
#CS OFF ALL Deselect all CS
#ACS OFF ALL
Deselect all ACS
```


## Programing Example

Linkage of coordinate systems

```
N10 #CS ON[3]
N20 #CS ON[4]
N30 #CS DEF ACT[5] (Store CS[3] O CS[4] in CS[5])
N31 #CS OFF ALL (Deselect all CS)
N32 #ACS ON[3]
N33 #ACS ON[4]
N34 #ACS DEF ACT[5] (Store ACS[3] O ACS[4] in CS[5])
N35 #ACS OFF ALL (Deselect all ACS)
N36 X0 Y0 Z0
N360 #CS ON [5]
N370 #ACS ON[5]
N380 #CS DEF ACT[1] (Store ACS[5] o CS[5] in CS[1])
N390 #ACS OFF ALL
N400 #CS OFF ALL
N500 #CS ON (Select CS[1])
N510 #CS OFF
M30
```


### 17.7. $\quad$ Define/activate a basic coordinate system (\#BCS)

## Release Note

This function is available as of CNC Build V3.1.3079.36.

The purpose of the basic coordinate system (BCS) is to compensate for offsets from the basic system.

Defining and storing a BCS:
\#BCS DEF [ [<BCS-ID>] ] [ <v1>,<v2>,<v3>,< $\langle\varphi 1>,<\varphi 2>,<\varphi 3>$ ]
(modal)
Define and store with simultaneous activation:
\#BCS ON [ [ $<B C S-I D>]$ ] [ <v1>,<v2>, <v3>, < $1>,<\varphi 2\rangle,<\varphi 3>$ ]

| \#BCS ON $[<B C S-I D>]$ | Select a stored BCS | (modal) |
| :--- | :--- | :--- |

\#BCS ON $\quad$ Select last BCS defined $\quad$ (modal)
\#BCS OFF Deselect last activated BCS (modal)
Parameter BCS-ID may not be programmed here since it is only permitted to deselect the last BCS activated.
$<B C S$-ID $>$ Coordinate system ID. The BCS-ID is assigned the default value 1 at program start. If the BCS-ID is not programmed with \#BCS DEF or \#BCS ON, the next free BCS-ID is calculated automatically. However, a BCS of this type is not longer available after it is deselected with \#BCS OFF!
<vi> Components of the translatory offset vector in [mm, inch]. (These refer to the main axes in the sequence contained in G17).
$<\varphi i>\quad$ Angle of rotation in [ ${ }^{\circ}$ ].

The BCS is modal and may be selected and deselected independently of a CS/ACS.
Zero offsets and reference point offsets may be programmed in the BCS. However, these values are only valid until the BCS is deselected and are not saved.


Fig. 180: Linkage with basic coordinate system \#BCS

### 17.7.5 Effector coordinate system (\#ECS ON/OFF)

## Notice

This function is not included in the scope of the standard license.
The use of this function requires a license for the "Transformations" extension package.

The effector coordinate system is mainly used to execute a withdrawal strategy after tool breakage, NC reset or program abort when machining takes place with a tool in any alignment. The ECS is determined by reversing the command TOOL ORI CS (Section Orienting tool (\#TOOL ORI CS) [ 718]). Instead of aligning the tool on the machining place, the machining plane is determined here perpendicular to the tool axis.

| \#ECS ON | Select ECS | (modal) |
| :--- | :--- | :---: |
| \#ECS OFF | Deselect ECS | (modal) |

No other coordinate system (CS) may be active when ECS is activated.
The ECS is then determined from the positions of the alignment axes so that its $Z$ axis is in parallel to the current tool axis. The position of the $X$ and $Y$ axes are then undefined (arbitrary) and must therefore be predefined internally. The origin of the ECS is generally located outside of the tool tip or tool axis, i.e. a collision-free tool withdrawal is only guaranteed by relative path motions along the effector $Z$ axis.


Fig. 181: Machining in a slanting hole
kernel Industrielle Steuerungstechnik GmbH

## Programing Example

## Effector coordinate system (ECS)

```
N01 #TRAFO ON (Select kinematic)
N05 #CS ON[1.5,0,32,14.5,0,45] (Select a BCS)
N10 #TOOL ORI CS
N15 X0 Y0 Z0
N20 LL TEILEPRG (Subroutine call for contour machining)
(Tool breakage, NC reset)
(Withdrawal strategy)
NO1 #TRAFO ON (Select kinematic)
N05 #ECS ON (Calculate the ECS
(according to the position of the orientation)
(axes)
N10 G91 G01 F200
N20 Z62 (Withdrawal motion along the tool or ECS-Z axis)
```

; . .
N400 M30

### 17.7.6 Temporary transition to the machine coordinate system (\#MCS ON/OFF)

## Notice

It is not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

The MCS functionality temporarily deactivates the active kinematics and/or Cartesian transformation as well as all offsets included in the axes in order to position machine axes directly. After leaving the MCS, the state before selection is restored.

For example, a tool change often requires the approach to a defined tool change position with known machine origin coordinates. Approaching this machine position may pose a problem in the CS since the CS axes are positioned by the NC program.

| \#MCS ON [ EX TOOL ] | Activate temporary transition to MCS | (modal) |
| :--- | :--- | :--- |
| \#MCS OFF | Deactivate temporary transition to MCS | (modal) |

EX TOOL A tool change in the MCS does not include tool offsets in order to permit direct positioning of the machine axes. This only takes place with \#MCS OFF.

The MCS has no restrictions regarding use of NC functionality. However, it is not possible to select RTCP/TLC , CS, ACS or ECS.
In addition, programmed offsets are only valid in the MCS until they are deselected and are not stored.

When the axis configuration is changed by external axis exchange (e.g. \#CALL AX..) in the MCS, it must be noted that a definite axis configuration is required to reactivate kinematic and/or Cartesian transformation.

## Programing Example

```
N10 #TRAFO ON
N20 #CS ON[1.5,0,32,14.5,0,45] (Activate a CS)
N30 G01 G90 F5000
N40 X0 Y0 Z0
N50 #MCS ON EX TOOL (Transition to the machine CS with the option)
        (EX TOOL' - Tool is only included)
        (with MCS OFF)
N60 LL WERKZWECHSEL (Subroutine call for tool change)
N70 #MCS OFF (Deactivate MCS, RTCP and CS are)
    (reactivated)
N100 #TRAFO OFF
N110 #CS OFF
N400 M30
(Transition to the machine CS with the option)
```


### 17.8 Auxiliary functions for coordinate transformation (\#WCS TO MCS, \#MCS TO WCS)

## Notice

Not included in the scope of the standard license. The use of this feature requires a license for the "Transformations" extension package.

On 5-axis machines and machines with non-Cartesian axis structure (e.g. hexapods), two typical variants are used to define path motions.

1st case: The user programs the contour in space with circles, straight lines or polynomials and the tool tip (TCP) is moved along the path depending on the programmed contour.

2nd case: The user programs the target point in space or workpiece coordinates (WCS) which are mapped onto machine coordinates (MCS). Depending on the maximum possible velocities in the axes, the TCP moves on a non-predictable curve (PTP). Due to the loss of TCP motion along a curve in space, the PTP motion is generally faster than the TCP path motion.

The above mentioned mapping of programmed WCS target points onto MCS target points (backward transformation) can be executed in the NC program by the following NC commands. The user must then calculate the approach of the calculated MCS target points explicitly in absolute dimensions.

For example, this method can be used to move each single axis sequentially out of a collision area.

## Attention

These commands only can be used with inactive transformation (\#TRAFO OFF) and when the coordinate system is inactive (\#CS OFF (ALL)).

## Calculation of machine coordinates (MCS) from workpiece coordinates (WCS):

```
#WCS TO MCS [ [CS<ID_expr>] [KIN] ] (non-modal)
```

$C S<I D$ expr $>\quad$ Calculate the target points taking into consideration an inactive Cartesian transformation with a specific valid ID.
KIN Calculate the target points taking into consideration the currently valid kinematic transformation which is however inactive.

The following axis-specific variables are provided to declare the WCS target points and store the calculated MCS target points. These variables permit write and read access:
V.A.WCS.* Access to the axis-specific auxiliary variable "workpiece coordinate" (WCS). The value does not
correspond to the programmed workpiece coordinates when transformation is active.
V.A.MCS.* Access to the axis-specific auxiliary variable machine coordinate (MCS). The value does not correspond to the programmed machine coordinates when transformation is inactive.

## Programing Example

## Auxiliary functions for coordinate transformations

The NC calculates the MCS target points using the Cartesian and kinematic backward transformation of specific WCS points. The user can access the calculated MCS positions using NC programming and reuse them in the NC program. The user defines the transformations which are to be executed by specifying the keywords and the associated CS IDs.

```
N02 #CS DEF[1] [0,0,0,0,0,45]
NOO #KIN ID [1]
N10 V.A.WCS.X=10
N20 V.A.WCS.Y=10
N30 V.A.WCS.Z=100
N40 #WCS TO MCS[CS 1, KIN] (Transformation of TCS in MCS)
N50 G00 Z=V.A.MCS.Z (Move single axes to MCS target points)
N60 G00 X=V.A.MCS.X
N70 G00 Y=V.A.MCS.Y
N..
:
```

    Incorrect use:
    N05 \#CS DEF[1][0,0,0,0,0,45]
N10 \#CS ON[1]
N20 V.A.WCS. $\mathrm{X}=100$
N30 V.A.WCS.Y = 0
N40 V.A.WCS.Z $=0$
N50 \#WCS TO MCS [CS 1] <- not permitted since a CS is active
NO5 \#KIN ID[12]
N10 \#TRAFO ON
N20 V.A.WCS. $\mathrm{X}=100$
N30 V.A.WCS.Y $=0$
N40 V.A.WCS.Z $=0$
N50 \#WCS TO MCS [KIN] <- not permitted since a transformation is active

## Calculation of workpiece coordinates (WCS) from machine coordinates (MCS):

The inverse mapping of MCS target points to WCS target points (forward transformation) is executed with the following command. For example, this method can be used to map measured values in the WCS (normally determined in the MCS).
\#MCS TO WCS [ [CS</D_expr>] [KIN] ]
(non-modal)
CS<ID_expr> Calculate the target points taking into consideration an inactive Cartesian transformation with a specific valid ID.
KIN Calculate the target points taking into consideration the currently valid kinematic transformation which is however inactive.

### 17.9 Auxiliary function to calculate motion limits in the workpiece coordinate system (\#GET WCS POSLIMIT)

The following command calculates the limits of a motion in the current workpiece coordinate system (WCS) in the direction of a programmed motion vector with the components (VC1, VC2, VC3). The vector components need not be specified in standardised form. The basis for path limiting are the axis-specific software limit switches. Based on these values, the controller calculates the motion limits in the current coordinate system.
\#GET WCS POSLIMIT [ VC1<expr> VC2<expr> VC3<expr> ]

VC1<expr>,
VC2<expr>,
Components of direction vector, REAL number
VC3<expr>

The following global variables read out the result of the calculation for the first three axes in the coordinate system.

| V.G.WCS_POSLIMIT_1 | Motion limit in the first main axis in WCS |
| :--- | :--- |
| V.G.WCS_POSLIMIT_2 | Motion limit in the second main axis in WCS |
| V.G.WCS_POSLIMIT_3 | Motion limit in the third main axis in WCS |

## Attention

The programmed motion direction must be absolutely retained for the real executed motion, otherwise the calculated motion limits are incorrect.
In the same way, no rotary axes may be programmed when kinematic transformation (\#RTCP) is active.

## Programing Example

Auxiliary function to calculate motion limits in the workpiece coordinate system

```
NO5...
N10 G98 X-100 Y-100 Z-100 (Shift negative software limits)
N20 G99 X200 Y200 Z300 (Shift positive software limits)
N30 #ECS ON (Select effector coordinate system)
N40 # GET WCS POSLIMIT [VC1=0,VC2=0,VC3=1] (Calculate WCS motion limit)
N50 G01 G90 Z=V.G.WCS_POSLIMIT_3 F2000 (Approach WCS motion limit in Z)
N60 #ECS OFF - (Deselect effector coordinate system)
N70...
Example of correct use of the command
N05...
N10 #CS ON [0,0,0,0,0,45]
N20 #GET WCS POSLIMIT [VC1=1 VC2=1 VC3=0]
N25 G01 G90 F2000
N30 X=V.G.WCS_POSLIMIT_1 Y=V.G.WCS_POSLIMIT_2 Z=V.G.WCS_POSLIMIT_3
N40 #CS OFF
N50...
```

Wrong, resulting motion direction does not correspond to setting:
N05...
N10 \#CS ON $[0,0,0,0,0,45]$
N20 \#GET WCS POSLIMIT [VC1=1 VC2=1 VC3=0]
N25 G01 G90 F2000
N30 $\mathbf{X}=\mathrm{V} . \mathrm{G} . \mathrm{WCS}$ _POSLIMIT_1 $\mathbf{Y}=\mathrm{V} \cdot \mathrm{G} . \mathrm{WCS}$ _POSLIMIT_2
N40 Z=V.G.WCS_POSLIMIT_-3
N50...

Positive software limit Z


Negative software limit Z

Fig. 182: Example of motion limits in the ZX plane in the current WCS

### 17.10 Orientation programming

Tool orientation can be programmed in a number of different ways. Basically, the representation of orientation depends on the basic kinematic (5-axis or robot kinematics) and on the settings of the CAD/CAM systems used.
In a conventional case, Cartesian coordinates are programmed in point/Euler angle representation for 5 -axis machining, i.e. for a 5 -axis machine with a CA tool head, i.e. the positions via $\mathrm{X}, \mathrm{Y}$ and $Z$ and machine angles for orientation via the configured axis identifiers C 1 and A 1 .

## Example:

## X50 Y50 Z100 C1=45 A1=45

Since the contours of a workpiece are normally represented in vectors in CAD/CAM systems, it is also normal to generate NC programs in vector representation. This means that the contour description is independent of the machine or kinematic structure.

Tool orientation is defined by a vector which is aligned by the tool tip (TCP) in the direction of the tool clamp. Direction vector components are always programmed by A, B, C (or I, J, K). So, when vector programming is active, tracking axes configured with the same name cannot be programmed due to unambiguity reasons.

With 5-axis kinematics, vector components are defined by the "virtual" axis identifiers $A, B$ and $C$. Example:
X50 Y50 Z100 A-0.5 B0.5 C0.7071


Fig. 183: Orientation vector at 5 -axis head

With 6-axis kinematics (e.g. robots), vector components are also defined by the "virtual" axis identifiers $A, B$ and $C$ or by the axis identifiers $I, J$ and $K$ which are also used in robotics.


Fig. 184: Orientation vector on robot

Example:
X-17.083 Y29.630 Z10 A-0.17083 B0.29630 C0.93969 or
X-17.083 Y29.630 Z10 I-0.17083 J0.29630 K0.93969

In order to correctly evaluate tool orientation, the appropriate orientation mode must be used and the corresponding transformation must be activated.

### 17.10.1 Programming and configuration of 5 -axis kinematics (\#ORI MODE)

Evaluation of point-vector programming after selecting the transformation is enabled by the NC command \#ORI MODE[..]. In conventional point-vector representation, the VECTOR_2DOF (2 Degrees Of Freedom) mode is set. It remains valid until program end (M30) or until another programmed change occurs.

The NC command has the following syntax:

## \#ORI MODE [ VECTOR_2DOF ]

VECTOR_2DOF $\quad \mathrm{A}, \mathrm{B}$ and C are components of the direction vector. The address letters $\mathrm{A}, \mathrm{B}, \mathrm{C}$ must always be used; they have no reference to configured axis names in the channel list. The vector components need not be specified in standardised form.
\#ORI MODE [VECTOR_2DOF] causes the preselection of orientation programming. Only when the transformation is active (\#TRAFO ON) are point-vector representations detected and evaluated.

The following NC commands revert to conventional orientation programming:

## \#ORI MODE [ ANGLE ]

ANGLE $\quad$ Angle values via configured axis names (default)

Special features relating to active coordinate systems (CS):

- With 5-axis RTCP transformations (incomplete) and an active CS, orientation is always represented independent of P-CHAN-00247.
- With complete 5-axis transformations and an active CS, orientation is represented dependent on P-CHAN-00247.
- Vector programming is also permitted with tube machining. Virtual axis programming may not be active (see [FCT-M5, Kin-ID $90-$ HD14 = 0]).

Alternatively, point-vector programming can be preconfigured with P-CHAN-00177. With ori.mode the user defines whether the meaning of values programmed with $A, B, C$ are read as normal coordinates or angle values with an active kinematic transformation or whether they are interpreted as corresponding vector components.

Alternatively, the following identifiers must be configured:
ori.mode ANGLE Angle values by configured axis names (default)
ori.mode VECTOR_2DOF Vector components by A, B, C
If orimode is unassigned, the default setting is active for orientation programming (orientation specified by rotation angle).

## Programing Example

Switching over orientation programming to point-vector representation

```
%example_1
:
#KIN ID [9]
:
#ORI MODE [VECTOR_2DOF]
#TRAFO ON
G01 F1000
X79.993 Y57.197 Z-39.993 A0.67520 B0.29702 C-0.67520
X79.973 Y57.392 Z-39.973 A0.66945 B0.32198 C-0.66945
X79.941 Y57.586 Z-39.941 A0.66316 B0.34705 C-0.66316
:
X79.255 Y58.978 Z-39.255 A0.58988 B0.55144 C-0.58988
X79.121 Y59.121 Z-39.121 A0.57735 B0.57735 C-0.57735
X78.903 Y59.319 Z-38.903 A0.55691 B0.61620 C-0.55691
X78.666 Y59.493 Z-38.666 A0.53439 B0.65487 C-0.53439
X78.414 Y59.643 Z-38.414 A0.50964 B0.69321 C-0.50964
X75.000 Z-35.000 A0.00000 B1.00000 C0.00000
:
#TRAFO OFF
M30
```

Programing Example

## Toggling between point-vector and point-angle representation

```
%example_2
:
#KIN ID [9]
:
#ORI MODE [VECTOR_2DOF]
#TRAFO ON
G01 F1000
X79.993 Y57.197 Z-39.993 A0.67520 B0.29702 C-0.67520
X79.973 Y57.392 Z-39.973 A0.66945 B0.32198 C-0.66945
X79.941 Y57.586 Z-39.941 A0.66316 B0.34705 C-0.66316
X79.255 Y58.978 Z-39.255 A0.58988 B0.55144 C-0.58988
X79.121 Y59.121 Z-39.121 A0.57735 B0.57735 C-0.57735
X78.903 Y59.319 Z-38.903 A0.55691 B0.61620 C-0.55691
X78.666 Y59.493 Z-38.666 A0.53439 B0.65487 C-0.53439
X78.414 Y59.643 Z-38.414 A0.50964 B0.69321 C-0.50964
X75.000 Z-35.000 A0.00000 B1.00000 C0.00000
:
#TRAFO OFF
#ORI MODE [ANGLE]
#TRAFO ON
G01 F1000
X10 Y10 Z10 C90 A15
X20 Y10 Z10 C90 A30
:
#TRAFO OFF
M30
```


### 17.10.2 Programming and configuration of 6-axis kinematics (robot) (\#ORI MODE)

The evaluation of point-vector programming after selecting the transformation is activated by the NC command \#ORI MODE[..]. In conventional point-vector representation, the VECTOR_ABC or VECTOR_IJK mode is used. It remains valid until program end (M30) or until another programmed change occurs. The behaviour of the fixed rotary axis is defined by 2 additional keywords.

The NC command has the following syntax:

```
#ORI MODE [ VECTOR_ABC | VECTOR_IJK FIXED_AX_IDX<idx> ]
```

or

## \#ORI MODE [ VECTOR_ABC | VECTOR_IJK TOOL_AX_IN_PLANE<id> ]

VECTOR_ABC

VECTOR_IJK I, J, K are components of the direction vector. The address letters I, J, K must always be used. Vector components need not be specified in standardised form. It is not permitted to use circle programming by I, J, K at the same time according to DIN 66025.

The 2 axes for the rotation angle to the tool orientation are obtained from the 3 vector components.. The angle setting of the third rotary angle is obtained from the joint angle settings at the time when the kinematic transformation is selected and remains unchanged during vector programming.
The axis index of the rotary axis not participating in orientation according to the Euler convention is obtained from considering the order of the axes that define the robot's position and hand orientation (see also description of P-CHAN-00178).
FIXED_AX_IDX<idx> Axis index of fixed rotary axis.

Example: Rotary axis C angle setting $45^{\circ}$ on selection, FIXED_AX_IDX = 5


As an alternative to the fixed rotary axis, it is also possible to define the plane (YZ, ZX ) containing either the $Z$ or the $Y$ tool axis. The third angle can then be determined so that the selected tool axis lies parallel to the defined plane at the target point (see also description of P-CHAN-00436).

TOOL_AX_IN_PLANE<id> Plane parallel to a tool axis.

Example 1: Tool axis $Z$ (red) parallel to the basic plane $Z X, T O O L \_A X \_I N \_P L A N E=1$


Basic plane ZX: Initial orientation


Orientation at target point

Example 2: Tool axis $Y$ (green) parallel to the basic plane $Y Z, T O O L \_A X \_I N \_P L A N E=2$


Basic plane YZ: Initial orientation


Orientation at target point
\#ORI MODE [VECTOR_...] causes preselection of orientation programming. Only when the transformation is active (\#TRAFO ON) are point-vector representations detected and evaluated

The following NC command reverts to conventional orientation programming:

## \#ORI MODE [ ANGLE ]

ANGLE
Angle values by configured axis names (default).

Special features relating to active coordinate systems (CS):

- With complete 6-axis transformations and an active CS, orientation is always represented dependent on P-CHAN-00247.

Alternatively, point-vector programming can be preconfigured with P-CHAN-00177. The ori.mode allows the user to define whether values programmed with A, B, C or I, J, K are read in the NC program as normal coordinates or angle values when the kinematic transformation is active or whether the values are interpreted as vector components.

Alternatively, the following identifiers must be configured:
ori.mode ANGLE Angle values by configured axis names (default)
ori.mode VECTOR_ABC Vector components by A, B, C
ori.mode VECTOR_IJK Vector components by I, J, K

If ori.mode is unassigned, the default setting is active for orientation programming (orientation specified by rotation angle).

The axis index of the fixed rotary axis is specified in P-CHAN-00178: ori.fixed_axis_index<idx> Axis index of fixed rotary axis

The plane parallel to the tool axis is specified in the channel parameter P-CHAN-00436: ori.tool_ax_in_plane<id> Plane parallel to tool axis plane

Specifications relating to the fixed rotary axis P-CHAN-00178 and the tool axis plane P-CHAN-00436 are mutually exclusive. If the two parameters are assigned, error ID 22027 is output when the controller starts up and the two values are corrected to zero.

## Programing Example

Switch over orientation programming to point-vector representation (ABC) and specify fixed rotary axis

```
%example_1
;...
#KIN ID [45]
; ...
#ORI MODE [VECTOR_ABC FIXED_AX_IDX=5]
#TRAFO ON
X50 Y50 A50 B0 C0
X75 Y150 Z180 A0 B0 C1
X149.316 Y150 Z180 A-0.0457 B0 C0.999
X149.316 Y150 Z165.012 A-0.0457 B0 C0.999
X150.0018 Y150 Z150.0279 A-0.0457 B0 C0.999
X162.1716 Y150 Z150.0621 A0.0349 B0 C0.9994
X172.1268 Y149.9997 Z149.3631 A0.1013 BO C0.9949
X178.7241 Y149.9997 Z148.5459 A0.1454 B0 C0.9894
X188.532 Y149.9997 Z146.7645 A0.2111 B0 C0.9775
X198.2064 Y149.9997 Z144.3474 A0.2758 B0 C0.9612
X207.7002 Y149.9994 Z141.2733 A0.3393 B0 C0.9407
X216.978 Y149.9994 Z137.5713 A0.4012 B0 C0.916
; ...
X150 Y150 Z180 A0.6111 B0.0014 C0.7916
X150 Y150 Z180 A0.0631 B0.0001 C0.998
X150 Y150 Z180 A0 B0 C1
;...
#TRAFO OFF
M30
``` Programing Example
Switch over orientation programming to point-vector representation (IJK) and specify the plane parallel to the tool axis

\footnotetext{
\%example 2
}
```

; ...
\#KIN ID [45]
;...
\#ORI MODE [VECTOR_IJK TOOL_AX_IN_PLANE=1]
\#TRAFO ON
X75 Y150 Z180 I0 J0 K1
X10.874 Y0 Z-29.875 I-.099 J0 K.995
X10.846 Y.666 Z-29.872 I-.099 J-.006 K.995
X10.667 Y1.976 Z-29.854 I-.097 J-.018 K.995
X10.464 Y2.748 Z-29.792 I-.095 J-.025 K.995
X10.208 Y3.429 Z-29.668 I-.093 J-.031 K.995
X9.879 Y4.075 Z-29.46 I-.091 J-.037 K.995
X9.517 Y4.713 Z-29.296 I-.088 J-.043 K.995
X9.126 Y5.328 Z-29.166 I-.085 J-.049 K.995
X8.285 Y6.492 Z-29.086 I-.077 J-.06 K.995
X7.387 Y7.597 Z-29.317 I-.068 J-.07 K.995
X6.9 Y8.108 Z-29.472 I-.063 J-.075 K.995
X6.385 Y8.598 Z-29.664 I-.058 J-.079 K.995
X5.825 Y9.038 Z-29.8 I-.053 J-.082 K.995
X5.218 Y9.412 Z-29.841 I-.047 J-.086 K.995
X3.924 Y10.011 Z-29.852 I-.035 J-.091 K.995
X2.56 Y10.43 Z-29.849 I-.023 J-.095 K.995
X1.182 Y10.657 Z-29.835 I-.01 J-.097 K.995
X.461 Y10.682 Z-29.769 I-.004 J-.098 K.995
X-.257 Y10.636 Z-29.639 I.002 J-.098 K.995
X-1.027 Y10.509 Z-29.435 I.009 J-.097 K.995
X-1.696 Y10.366 Z-29.297 I.015 J-.096 K.995
X-3.083 Y9.956 Z-29.106 I.028 J-.093 K.995
X-4.428 Y9.482 Z-29.258 I.041 J-.088 K.995
X-5.462 Y9.007 Z-29.478 I.05 J-.083 K.995
X-6.068 Y8.681 Z-29.657 I.055 J-.08 K.995
X-6.642 Y8.299 Z-29.782 I.061 J-.076 K.995
X-7.696 Y7.337 Z-29.826 I.07 J-.067 K.995
X-8.601 Y6.233 Z-29.831 I.078 J-.057 K.995
; ...
\#TRAFO OFF
M30

```

\subsection*{17.11 Status \& Turn (IS, IT)}

An option exists to specify the robot pose for the corresponding Cartesian position as an alternative to axis-specific positioning and to obtain a more precise specified position of a \#PTP movement for industrial robots.

The robot pose is described using 2 additional key figures (Status \& Turn).

\section*{Notice}

Robot positioning with Status \& Turn is currently only available for kinematic type 45.

\section*{Status bit}

The robot pose is divided into 3 criteria. If a criterion applies, a corresponding numerical value is added to the status.
1. criterion If the wrist is behind axis A1, decimal 1 or binary 1 is added (yellow area in the figure on the left).


Fig. 185: The intersection of the hand axes (arrowhead) is in the (blue) base area.

2nd criterion If the wrist is in front of the straight line through axes A2 and A3, decimal 2 or binary 10 is added (centre and right image).


Fig. 186: Status bit 1 for robots with an offset between axis A3 and axis A5

3rd criterion (legacy value, not normally used any more): Indicates the position of axis A5. If A5 \(>0\), decimal 4 or binary 100 is added.


Fig. 187: Status bit 2 for axis angle position \(A 4=0^{\circ}\) and \(A 4=180^{\circ}\).

\section*{Turn bit}

The turn value lists the negative signs of the axis angles.
When the turn value is considered in a binary representation, the sign of each axis angle is assigned to a bit. These are then added to a number, i.e. the turn.

If the axis angle of an axis is \(<0^{\circ}\), the value is 1 .
\begin{tabular}{|l|l|l|l|l|l|l|}
\hline & \(\mathrm{A} 6<0^{\circ}\) & \(\mathrm{A} 5<0^{\circ}\) & \(\mathrm{A} 4<0^{\circ}\) & \(\mathrm{A} 3<0^{\circ}\) & \(\mathrm{A} 2<0^{\circ}\) & \(\mathrm{A} 1<0^{\circ}\) \\
\hline Binary & 100000 & 010000 & 001000 & 000100 & 000010 & 000001 \\
\hline Decimal & 32 & 16 & 8 & 4 & 2 & 1 \\
\hline
\end{tabular}

If all 6 axis angles are in the negative range, this results in a turn value of decimal 63 or binary 111111; accordingly, this results in decimal 0 and binary 000000 for 6 positive axis angles.

\section*{Description}

The additional parameters Status \& Turn are available for unambiguous programming of the robot poses (Kin_Typ_45) with Cartesian target coordinates of a PTP movement.

Status \& Turn are programmed with the prefixes "IS" and "IT":
```

\#PTP ON
G... X.. Y.. ... IS.. IT..
\#PTP OFF

```

Binary numbers can be programmed with the following syntax:
'B<0...1>', or '2\#<0...1>', or '02\#<0...1>'.
When binary numbers are used, the syntax is as follows:
Status: IS'Bxxx'
Turn: IT'Bxxxxxx'
When decimal numbers are used, the syntax is as follows:
Status: IS<expr>
Turn: IT<expr>

\section*{Display values}

The following CNC objects are provided for this function:
- mc_st_valid_r: Validity of the status \& turn value (Task COM index group \(0 \times 12010<\mathrm{C}_{\text {ID }}>\) Index offset \(0 \times B 1\) )
- mc_st_status_r: Status value of kinematic 45 (Task COM index group \(0 \times 12010<\mathrm{C}_{\text {ID }}>\) Index offset \(0 \times B 2\) )
- mc_st_turn_r: Turn value of kinematic 45
(Task \(\overline{\mathrm{C}}\) OM index group \(0 \times 12010<\mathrm{C}_{1 \mathrm{D}}>\) Index offset \(0 \times B 3\) )

\section*{Programming examples}

If Status \& Turn are not programmed, the target point is determined on the axis angle level using the shortest way strategy.

\section*{Programing Example}

Status \& Turn programmed with binary numbers
```

N010 \#PTP ON
N020 G01 X1100 Y0 Z1400 A0 B90 C0 IS'B010' IT'B000010' F5000
N030 G01 X1200 ;target point is determined by shortest way
NO40 \#PTP OFF

```

\section*{Programing Example}

Status \& Turn programmed with decimal numbers
```

N010 \#PTP ON
N020 G01 X1000 Y0 Z1400 A0 B90 C0 IS2 IT2 F5000
N030 \#PTP OFF
N040 G01 X1500
N050 G01 Y1000
N060 G01 X-1000
N070 \#PTP ON
N080 G01 X-1000 Y-1000 ; target point is determined by shortest way
N090 \#PTP OFF

```
kernel Industrielle Steuerungstechnik GmbH

\section*{18 Programming modulo axes}

The default mode for modulo programming supports the specific definition of the direction of rotation, position by two signs and limiting to maximum one revolution in absolute dimensions


\section*{Attention}

The programming of 2 signs (direction and position) is only permitted if the axis has the "Modulo" axis mode P-AXIS-00015). Positioning is always executed on the shortest way if no sign is programmed directly after the axis name.
In addition, you can switch to a mode that always positions on the shortest path (Section Positioning on the shortest way [ 768]). In this mode, the use of two signs is also permitted. However, evaluation is based on the following rule:
\[
\begin{aligned}
& --=>+ \text { (minus minus is plus) } \\
& +-=>- \text { (plus minus is minus) } \\
& -+=>- \text { (minus plus is minus) } \\
& ++=>+ \text { (plus plus is plus) }
\end{aligned}
\]

\section*{Programming in absolute dimensions (G90):}
- The value assigned to the axes (target point) is shifted to the modulo range. Therefore, a maximum of one revolution can be moved.
- The value may be a numerical expression such as [3*2+5], P1, [P1+P2-3], [-30].
- The first sign of the value after the axis name always defines the direction of rotation. Every further sign is evaluated as a part of the (absolute) position definition.

\section*{Example (assuming: modulo range 0-360 \({ }^{\circ}\) )}

G90 G1 C+560 * G90 G1 C+200 (Move to position 200 in + direction)
G90 G1 C-P1 (Go to position P1 (with implicit modulo) in - direction)
- If programmed position = current position, no motion.
- The motion path of a modulo axis is not limited by software limits.

\section*{Programming in incremental dimensions (G91)}
- The value assigned to the axis indicates the amplitude of rotation of the axis with reference to the previous position. The first sign of the value after the axis name always defines the direction of rotation. Additional signs are not permitted in incremental programming.

\section*{Example (assuming: modulo range 0-360 \({ }^{\circ}\) )}

G91 G1 C+560 (Movement to "current position plus 560" in + direction)
- If the value is greater than the modulo value, the number of revolutions is taken into account. Therefore, a motion of more than one revolution is permitted.

The following V.A. variables permit read access to the current axis-specific modulo settings
\begin{tabular}{ll} 
V.A.MODE[i] & \begin{tabular}{l} 
delivers the axis mode according to the axis table. \\
e.g. 4 if the axis is the modulo type. \\
-> used to read modulo axes
\end{tabular} \\
V.A.MODULO_VALUE[i] & \begin{tabular}{l} 
used to read the modulo range \\
e.g. 360 for a modulo range of 0-360 \\
(this value is irrelevant with "non modulo axes").
\end{tabular}
\end{tabular}

\section*{Programing Example}

Programming examples of modulo programming in absolute dimensions


\(P 1=-350\)
G90 G1 C-P1 <=> G1 C-[-350] <=> G1 C-10 <=> Go to position 10 in-direction


G90 G1 C+450 <=> G1 C+[450 mod 360] <=> Go to position 90 in + direction


\section*{Programing Example}

\section*{Examples of correct programming:}
```

C+200 Rotate in positive direction to position 200
C-200 Rotate in negative direction to position 200
C+-200 Rotate in positive direction to position -200 (= +160)
C+[-200 Rotate in positive direction to position -200 (= +160)
C-200 Rotate in negative direction to position -200 (= +160)
C-[-200] Rotate in negative direction to position -200 (= +160)
C200 Rotate on shortest way to position 200
C[+200] Rotate on shortest way to position 200
C[-200] Rotate on shortest way to position -200

```

\section*{Programing Example}

\section*{Examples of incorrect programming:}

None because the first sign after the axis name determines the direction of rotation and every additional sign belongs to the position expression.

\section*{Programing Example}

\section*{Programming examples of modulo programming in relative dimensions}

G91 G1 C+30


G91 G1 C-30


\section*{Programing Example}

\section*{Examples of correct programming:}

C+200 Rotate in positive direction to "current position plus 200"

C-200 Rotate in negative direction to "current position minus 200"

C200 Rotate in positive direction to "current position plus 200"

\section*{Programing Example}

\section*{Examples of incorrect programming:}
\begin{tabular}{ll}
\(C+-200\) & Error: Negative motion path during rel. programming not allowed. \\
\(C-200\) & Error: Negative motion path during rel. programming not allowed. \\
\begin{tabular}{ll}
\(\mathrm{Cl}=-1\) \\
\(C-[P 1]\) & Error: Negative motion path during rel. programming not allowed.
\end{tabular}.
\end{tabular}

\subsection*{18.1 Positioning on the shortest way}

This mode always permits the positioning of modulo rotary axes on the shortest way. This obviates the need for programming signs.

This mode is activated by the parameter P-CHAN-00346. However, if a pre-definition of rotation direction is desired, this requires relative programming in this mode.

\section*{Programing Example}

\section*{Programming examples of modulo programming in absolute dimensions}
```

Assuming: Modulo range $0-360^{\circ}$, G90, current axis position $0^{\circ}$
C200 Rotate on shortest way (-160) to position 200
C+200 Rotate on shortest way (-160) to position 200
C-200 Rotate on shortest way (+160) to position -200 (= +160)

```
\(\qquad\)

\section*{Programing Example}

\section*{Programming examples of modulo programming in relative dimensions}

Assuming: Modulo range \(0-360^{\circ}\), G91, current axis position \(+60^{\circ}\)

C+200 Rotate in positive direction to position \(260(60+200)\)

C-200 Rotate in negative direction to position 220 (60-200)
kernel Industrielle Steuerungstechnik GmbH

\section*{19 Extended tool programming}

\subsection*{19.1 Description of function}

\subsection*{19.1.1 Tool ID}

Within the scope of extended tool programming, the CNC provides tool management tasks (WZM) with new tool-specific communication objects. This refers to the enhanced tool (WZ) descriptions and tool service life variables.

In the present standard programming, single-element numbers are used to identify tools in the NC program. According to DIN 66025, this numerical ID is programmed together with the D word that is used to include new data (computational tool change). In conjunction with the T language command, the ID defines the next tool that is to be physically changed.

To include new data, this data must be requested from the present external tool management. The tool management function has special manufacturer-specific algorithms that are used to determine the tool that must be changed based on tool identification. It must be taken into account that the transferred tool ID only defines the tool type and that the tool magazine may contain several tools of the same type (alternate tools) that are ready to be used. Therefore, a specific tool cannot be clearly identified in the NC program.

In the first instance, the T number is used as technological information. This means that it reaches the PLC over the NC channel. According to DIN 66025, M06 is used to trigger the physical insertion of the new tool into the working spindle. By specifying the two commands "T with tool number" and M06 separately, preparatory measures can be taken (in the tool magazine, for example) after the T command and before M06 actually inserts the tool into the working spindle.

In the extended tool management function, tool data is identified by a three-digit tool number. Tool ID number:

Tool ID = base tool number + alternate tool number + modification number

The base tool number describes the tool type and the alternate tool number describes a tool unit of this type. The modification number has a purely data-related significance. It permits the use of various data records or a tool.

\subsection*{19.1.2 Tool life data recording}

When tool life values are recorded, the contact time (service life) and the distance covered by the tool during contact (service distance) must be calculated. These channel parameters are required for the configuration of the tool life data recording function.
Only travel by motion blocks is considered in the basic setting. Positioning by rapid traversing has no impact on the tool life quantities.

The interpolator displays the data (tool ID, contact time, contact distance) after the tool is replaced. The trigger point for activating tool life data recording can be linked with by P-CHAN-00482 using the T or the D word.

The service life is recorded in ms ; the service distance is recorded in mm .
Tool life data recording can be adapted to tool use by means of weighting factors [ \(>711\) ].

\subsection*{19.2 Programming commands and variables (V.TOOL.) (\#TOOL DATA, \#TOOL PREP)}

The tool ID is programmed with plain text commands. A request for new tool data with \#TOOL DATA corresponds to the D command; the preparatory technology command for physical tool change with \#TOOL PREP corresponds to the T command.
\#TOOL DATA [<basic> [, <sister> [, <variant> ] ] ] (Request new tool data)
\#TOOL PREP [<basic> [, <sister> [, <variant> ] ] (Announce a tool change)

The number of parameters is fixed [6] [ \(>\) 819]-9.18. Expedient values are between 1 and 3 . If two parameters are expected, they are the base tool number and the alternate tool number.

The base tool number must absolutely be specified (basic). In addition, it the number of parameters is assigned the value 3 , sister can be programmed for the alternate tool number and optionally variant for the modification. If sister or variant are not programmed (a comma follows a comma or a closing bracket follows a closing bracket), zero is inserted instead.

\section*{Programing Example}

Programming commands and variables with 'sister' and 'variant'
```

\#TOOL DATA [ P10, "SISTER", 0 ] < = > \#TOOL DATA [ P10, "SISTER", ]
and
\#TOOL PREP [P10, 0, "VARIANT"] < = > \#TOOL PREP [P10, , "VARIANT"]

```

The mathematical expressions used to specify the \(D\) and \(T\) commands are to be interpreted as a base tool number. Therefore, the tool management function retains the same degree of freedom as before to select the tool data record.

T<basic> or D<basic>

Decoder variables are used to implement access to the elements of tool identification. The current software version already provides the option to interrogate the number of the last tool requested from the external tool management function via V.G.T_AKT. For compatibility reasons, this variable is retained. In parallel to the new syntax introduced below, it always represents the base tool number.
\begin{tabular}{|ll}
\hline V.TOOL.BASIC & (Read access to the last base tool number programmed \\
V.TOOL.SISTER & (Read access to the last alternate tool number programmed) \\
V.TOOL.VARIANT & (Read access to the last modification number programmed) \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Programming commands and variables with 'sister’}
```

\#TOOL DATA [ P10, "SISTER", 3 ]
\#TOOL PREP [V.TOOL.BASIC, V.TOOL.SISTER, V.TOOL.VARIANT]

```

\subsection*{19.2.1 Weighting factors for tool life and tool life distance ((V.TLM)}

The NC program can change the weighting of tool life values. The changeable factors are used to adapt tool life data recording to tool use.
If every tool change is initiated by the T or D command (see P-CHAN-00482), the complete tool ID, the service life and the service distance are sent to the tool management automatically. All parameters are then zeroed and tool life data recording is enabled for the new tool substituted.
The following two decoder variables are used to program the weighting factors of service life and service distance (access is not synchronous with real time):
\begin{tabular}{ll} 
V.TLM.TIME_FACT & (Weighting of service life) \\
V.TLM.DIST_FACT & (Weighting of service distance) \\
\hline
\end{tabular}
V.TLM.TIME_FACT
(Weighting of service distance)

The variables can be read and written. Both factors are \(100 \%\) at program start. Both variables may be written in an NC block.

\section*{Example}

A tool that is always in contact should be weighted with a factor of \(100 \%\). However, if material is only removed along half the motion path, a weighting factor of 0.5 can be included in the calculation. The default value of the weighting factors of service time and service distance is 1.0 .

\section*{Acquisition conditions:}
- Rapid traverse blocks are not included in tool life acquisition.
- Tool life acquisition is stopped at a feedrate of zero.
- With the exception of rapid traverse interpolation, all motion types are included in tool life acquisition. For example, G01, G02, G03, spline interpolation and G63 are included.
- The weighting factors are included in the calculation.
- A distinction is not made between main and tracking axes for axes participating in the motion. The path feedrate is always used to add up the distance. In the case of tracking axes programmed on their own in the block, the path covered by the tracking axis is added to the service distance. If this is not desired, the programmer can correct it by specifying the weighting factors V.TLM.TIME_FACT/DIST_FACT \(=0\).
- Active master/slave arrangements are not taken into account.
- In the case of a reset or program abort, the last current values are also saved to the database of the tool management function.
- If only substitution occurs, i.e. no tool was previously in the work spindle and the current T number is zero, no data is sent.
- Tool data is only sent if a tool management system is actually present (P-CHAN-00016).

\subsection*{19.2.2 Reading/removing tool life values (\#TOOL LIFE READ/REMOVE)}

The following commands are provided for access to tool life data before the next tool change. For example, they are required to determine tool life data for tools in slave axes in synchronous mode since this data is not acquired as described in Section Weighting factors for tool life and tool life distance ((V.TLM) [> 771]. The following command reads the tool life data of the active tool in the current channel and outputs this data to the tool management function (WZM) for any tool (e.g. tool ID of a slave axis).
\#TOOL LIFE READ [<basic> [, <sister> [, <variant> ] ] (Read current tool data and assign (a tool ID in the WZM)

The same rules as e.g. for \#TOOL PREP (Section Programming commands and variables [ 770]) apply to programming tool ID. According to the HÜEMNOS convention, a tool ID can include 1,2 , or 3 tool parameters. It is mandatory to specify at least one parameter which is always interpreted as the base tool number (basic). Tool parameters may be any mathematical expression; z.B. V.G.T_AKT, V.TOOL.BASIC, Pxx, V.L.xxx etc.
After sending the current tool life data to the WZM, internal tool life acquisition is not reset. Adding continues.

The following command resets to zero the previously acquired tool life data of the tool currently active in the current channel without sending the data to the WZM.

\section*{\#TOOL LIFE REMOVE (Delete current tool life data)}

\section*{Programing Example}

\section*{Reading and deleting tool life data}
```

:
.... Normal mode
:
\#TOOL LIFE READ [V.G.T_AKT]
\#TOOL LIFE REMOVE
Select synchronous mode
:
.... Synchronous operation
:
Deselect synchronous mode
\#TOOL LIFE READ [T10]
:
.... Normal mode
:

```
(Read tool life data of current tool)
(Reset tool life data for separate)
(acquisition of tool life data during)
(synchronous operation)
(Assumption: T10 is also moved in synchronous)
(mode --> Send tool life data)
(to WZM for T10)

\subsection*{19.2.3 Refreshing tool data (\#TOOL REFRESH)}

Following command is provided for the repeated inclusion of tool data of the active tool, e.g. in additional axes after an axis exchange:

\section*{\#TOOL REFRESH}
(Refresh data of active tool)

The command results in the immediate inclusion of the already available data of the active tool. No more tool data is adopted from an internal tool list [5] [> 819] or requested from an external tool management function. The D word reprogramming function can be deselected.

\section*{20 Positioning axes}

A complete list of axis-specific additional functions is contained in the overview of commands in the Appendix under Additional axis-specific functions (<X>[..]) [> 816].

Positioning axes are translatory or rotary axes which can be interpolated in the same NC channel independent of the path axis compound. Each positioning axis has its own axis interpolator and can be commanded at its own feedrate. Rotary positioning axes in modulo mode always move on shortest way.

\section*{Restrictions:}

An axis cannot be programmed as a positioning axis if:
- This axis is currently moving in synchronous mode
- A kinematic or Cartesian transformation is active and certain conditions are not fulfilled (see Section Cartesian / kinematic transformation and positioning axes [ 782]).
- Block search is active
- Simulation modes (Online-Simulation, Contour visualisation, Machining time calculation) are active
- Spline interpolation is active
- Polynomial contouring is active
- Turning functions are active

\subsection*{20.1 Independent axes (INDP_SYN, INDP_ASYN) (\#WAIT INDP, \#WAIT INDP ALL)}

Two different operation modes are provided to program independent axes:
- Command value based synchronisation of path axes and independent axes at block end.
- Command value based synchronisation of path axes and independent axes over several blocks.


Fig. 188: Motion diagram of path axis compound/independent axes

\section*{Notice}

No offsets are considered for independent axes.

Additive manual mode (G201/G202) with an independent axis is possible.

Programming syntax for independent axes:
```

<Achsname> [ INDP_SYN | INDP_ASYN G90 | G91 G00 | [G01 | G100 FEED=.. |
TIME=.. | FEED_MAX_WEIGHT=..] POS=.. [SLOPE_TYPE=..]
{M..} {H..} [DRY_RUN] [ACC_WEIGHT=..] [RAPID_ACC_WEIGHT=..]{\}]

```
\(\left.\begin{array}{ll}\text { <Achsname> } & \begin{array}{l}\text { Name of independent axis } \\ \text { INDP_SYN }\end{array} \\ & \begin{array}{l}\text { Identifier for synchronous (blockwise) independent axis motion. The transition to the } \\ \text { next block is only executed if all axes have reached their target point. Must always be }\end{array} \\ \text { programmed as the first keyword. }\end{array}\right]\)
(*) only possible with synchronisation modes MOS, MVS_SVS, MVS_SNS, MNS_SNS.

Axis-specific M/H functions can also be output to an independent axis without programming a motion. This only requires the identifier INDP_SYN or INDP_ASYN.
```

<axis_name> [ INDP_SYN | INDP_ASYN M.. {M..} H.. {H..} { \ } ]

```
<axis_name>
INDP_SYN/INDP_ASYN
M<expr>
H<expr>
1

Name of independent axis
Identifier for an independent axis
Axis-specific M function
Axis-specific H functions
Separator ("backslash") for clear programming of the command over multiple lines.

Command value based synchronisation of specific asynchronous axis motions (INDP_ASYN) can be forced by the NC command:
\#WAIT INDP [ <Achsname> \{ ,<Achsname> \}]
<Achsname> Name of asynchronous axis

\section*{Notice}

If an asynchronous axis is again programmed with a motion before or without a corresponding \#WAIT INDP [ ], command value based synchronisation is implicitly executed in the interpolator.

Command value based synchronisation of all currently active asynchronous axis motions (INDP_ASYN) is forced by the NC command:

\section*{\#WAIT INDP ALL}

\section*{Notice}

If the axis of a pre-assigned axis-specific M/H function (P-CHAN-00039, P-CHAN-00025) is programmed in the same NC block as an independent axis, an error message is output.
Example: M 10 is pre-assigned for a specific X axis (m_default_outp_ax_name[10] x ):
N10 M10 X [INDP_SYN G01 G90 POS10 FEED1000 M7]
|___ < Error!

\section*{Programing Example}

\section*{Independent axes}
\begin{tabular}{|c|c|c|}
\hline N10 & X10 Y11 Z[INDP_SYN & ```
POS50 G01 FEED100 G90] (N10 ends when X,Y +)
(the indep. synchr. Z axis)
(end their motions)
``` \\
\hline N20 & X20 Y22 & (N20 is executed after all motions in) (N10 have ended) \\
\hline N30 & X5 Y10 Z[INDP_ASYN & ```
POS500 G01 FEED200 G90] (N30 ends when X + Y)
    (end their motions;)
(the independent asynchronous)
(Z axis continues its motion)
``` \\
\hline N40 & X20 Y30 & (N40 is interpolated; the asynchronous independent) (Z axis continues its motion) \\
\hline N50 & \#WAIT INDP[Z] & (Forced synchronisation of the \(Z\) axis: wait) (until target position \(Z 500\) from N30 is reached) \\
\hline N60 & X30 Y40 Z60 & (Interpolation in N60 with X,Y, Z in coordinated motion) (starts after synchronisation takes place in N50) \\
\hline N70 & Z[INDP_SYN M50] & (Output of M50 via independent Z axis) \\
\hline N80 & & \\
\hline
\end{tabular}

\section*{Example 2:}


\section*{Example 3:}
```

%dry_run
N100 X1 Y2 Z3 ;IPO=3, LR=3, offset=0
N200 G01 X10 F100 Z[INDP SYN POS=4 G01 G90 \
FEED=120 DRY_RUN] ;IPO=4, LR=3, offset=1
N300 Y20 F10000
N350 Z[INDP_SYN POS=7 G00 G90] ;IPO=7, LR=6, offset=1
N360 Z[INDP_SYN POS=4 G01 G91 \
FEED=100 DRY_RUN] ;IPO=11, LR=6, offset=5
;Remove DRY RUN offset
NOO1 \#TIME 2
N111 \#CHANNEL INIT[CMDPOS] ;IPO=6, LR=6, offset=0
N222 \#TIME 2
N400 Y10 Z5
M30

```

\subsection*{20.2 Oscillating axes (OSC)}

\section*{Release Note}

The availability of this function depends on the configuration and on the version scope.

An oscillating axis motion is required in certain machining technologies, e.g. grinding, and this is executed mainly independently of a path motion.

This motion referred to below as an "oscillating motion" is executed by the tool with periodic reversal across the workpiece.
An oscillating axis in grinding is presented below as an example. The workpiece is machined by superimposing the oscillating X motion on positioning motions in the Y and Z axes.


Fig. 189: Grinding with an oscillating axis

The essential characteristics of the oscillating motion result from the oscillating motion between two absolute positions as well as the feedrate
The oscillating motion start and stop and its parameters are defined in the NC program
Any axis can be defined as an oscillating axis within the scope of the configured axes. The oscillating motion is asynchronous to the path motion.
The oscillating motion is deactivated either:
- directly by an NC command
- or implicitly by programming a path motion for the oscillating axis
- or implicitly by requesting axis positions to synchronise decoding and interpolation
- or implicitly at the end of the NC program

The type of velocity profile can be defined in the dynamic phase by specifying the slope type in the channel parameters P-CHAN-00071 (linear/non-linear slope) for the oscillating motion.
The programming syntax is based on the axis-specific programming of independent axes. After the axis identifier, the parameters of the oscillating motion are defined via keywords and, if applicable, an associated value:
```

<axis_name> [ OSC ON | OFF FEED<expr> | FREQ<expr> | TIME<expr>
[1ST_POS<expr> 2ND_POS<expr>]|[ZERO_POS<expr> EXCUR<expr>]
[1ST_DELT<expr> 2ND_DELT<expr>] [NBR_OSC<expr>] {\}]

```
\begin{tabular}{ll} 
<axis_name> & \begin{tabular}{l} 
Name of the oscillating axis \\
OSC
\end{tabular} \\
Identifier for "Oscillating" function. Must always be programmed as the first keyword. \\
ON & \begin{tabular}{l} 
Oscillation on. The motion is stopped at block end when a path motion is active and \\
the oscillating motion is then commanded. \\
Oscillation off. The oscillating axis can then move again in the coordinated motion. If \\
the oscillating motion is stopped implicitly if it is not previously deselected when a \\
new axis motion is programmed.
\end{tabular} \\
OFF & Feedrate of the oscillating motion in [mm/min, \(\mathrm{m} / \mathrm{min}\), inch/min] \\
FEED<expr> & Frequency of the oscillating motion in [Hz] \\
FREQ<expr> & Period of the oscillating motion in in [s] \\
TIME<expr> & First reversal position in [mm, inch] \\
1ST_POS<expr> & Second reversal position in [mm, inch] \\
\(2 N D \_P O S<e x p r>\) & Zero point or zero crossing of the oscillating motion in [mm, inch] \\
ZERO_POS<expr> & Excursion in [mm, inch] \\
EXCUR<expr> & Wait time at first reversal position in [s] \\
\(1 S T \_D E L T<\) expr> & Wait time at second reversal position in [s] \\
\(2 N D \_D E L T<e x p r>\) & Number of oscillations \\
NBR_OSC<expr> & Separator ("backslash") for clear programming of the command over multiple lines.
\end{tabular}

The characteristic of the oscillating motion is determined by the locations of the reversal positions and the axis feedrate. The reversal positions can be either specified directly or alternatively they are determined automatically via the zero position and the excursion.

Oscillating positions are always absolute positions.
After an oscillating motion is deselected, the tool always stops at oscillating position 2.

Alternatively, the oscillating velocity can be determined by feedrate, frequency or period.

If no restriction occurs due to the dynamic axis characteristics, the frequency and the period are maintained exactly when the linear slope is used and maintained approximately when the non-linear slope is used.


Fig. 190: Positioning procedure with pendulum movement

\section*{Programing Example}
```

Specifying the oscillating motion path via reversal positions
N10 X[OSC ON 1ST_POS=-100 2ND_POS=100 FEED=1000]
Specify the oscillation travel distance via the zero position and the excursion,
10 oscillations
N20 X[OSC ON ZERO_POS=0 EXCUR=100 FEED=1000 NBR_OSC=10]
Specify 1 Hertz oscillation frequency
N30 X[OSC ON ZERO_POS=0 EXCUR=100 FREQ=1]
Specify a 4s oscillating period
N40 X[OSC ON ZERO_POS=0 EXCUR=100 TIME=4]
Oscillating motion with feed motion of a path axis
N50 X[OSC ON 1ST_POS=111 2ND_POS=222 FREQ=1]
N60 G01 G90 Y500 F200
Oscillate with wait times of 0.5 s each at reversal positions
N70 X[OSC ON 1ST_POS-100 1ST_DELT0.5 2ND_POS200 2ND_DELTO.5 FEED1000]
Deselect oscillation
Oscillation is stopped when reversal position 2 is reached:
N80 X[OSC OFF]
Fast oscillation stop
If a feedrate is programmed with FEED in combination with OFF, the oscillating motion
is stopped immediately (feedhold of oscillating axis) and the reversal position 2 is
directly approached at the new feed rate.
N90 X[OSC OFF FEED=5000]

```

Notice
The equals signs between the keyword and the value are optional.

\section*{20.3 \\ Cartesian/kinematic transformation and positioning axes}

\subsection*{20.3.1 Positioning and shifts}

The user must program absolute positions for this axes in the case of conventional operation and in CS mode (\#CS, \#ACS) with active independent axis or oscillating axis. This means that, if a tool change was executed, the tool length my have to be taken into consideration when programming the axes. Zero offsets (G54...G59) or reference point offsets (G92) previously programmed are not effective.

When kinematic transformation (\#TRAFO) is active, tool offsets are considered directly in the transformation. This means that they are also considered for active independent axes or oscillating axes. In this case too, previously programmed zero offsets (G54...G59) or reference point offsets (G92) are not effective.

\subsection*{20.3.2 Restrictions}

An oscillating motion or independent axis motion must be deselected before selecting a new Cartesian or kinematic transformation.

A positioning axis may only be programmed:
- with Cartesian kinematics and
- only in the 3rd axis (generally A. \(Z\) axis) for tools perpendicular to the XY machine base (e.g. A axis to \(0^{\circ}\) with CA head kinematic).

\section*{Programing Example}

Programming independent axes:
```

N10 \#KIN ID[9]
N20 \#TRAFO ON
N30 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N40 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N50 G01 G90 X100 F0.1
N60 \#TRAFO OFF
N100 M30

```
```

N10 G00 X0 Y0 Z0 C0
N20 \#CS ON [0,0,0,0,0,45]
N30 Z[INDP ASYN G01 G90 POS20 F0.01 SLOPE TYPE=STEP]
N40 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N50 G01 G90 X100 F0.1
N60 \#CS OFF
N100 M30

```
```

N10 \#KIN ID[9]
N20 \#TRAFO ON
N30 \#CS ON[0,0,0,0,0,45]
N40 Z[INDP ASYN G01 G90 POS20 F0.01 SLOPE TYPE=STEP]
N50 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N60 G01 G90 X100 F0.1
N70 \#CS OFF
N80 \#TRAFO OFF
N100 M30

```

\section*{Programing Example}

Programming oscillating axes:
```

N10 G00 X0 YO ZO C0
N20 \#KIN ID[9]
N30 \#TRAFO ON
N40 Z[OSC ON 1ST POS=10 2ND POS=20 FEED=1.00]
N50 G01 G90 X100 Y100 F0.1
N60 Z[OSC OFF FEED=2.00]
N70 \#TRAFO OFF
N100 M30

```
N10 G00 X0 YO ZO CO
N20 \#CS ON \([0,0,0,0,0,45]\)
N30 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N40 G01 G90 X100 Y100 F0.1
N50 Z[OSC OFF FEED=2.00]
N60 \#CS OFF
N100 M30
N10 G00 X0 YO ZO C0
N20 \#KIN ID[9]
N30 \#TRAFO ON
N40 \#CS ON \([0,0,60,0,0,45]\)
N50 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N60 G01 G90 X100 Y100 F0.1
N70 Z[OSC OFF FEED=2.00]
N80 \#CS OFF
N90 \#TRAFO OFF
N100 M30

\section*{Programing Example}

\section*{Impermissible nested programming}

The following program extract shows an impermissible nesting of CS with kinematic transformation and oscillation:
```

N10 \#KIN ID[9]
N20 \#TRAFO ON
N30 Z[OSC ON 1ST POS=10 2ND POS=20 FEED=1.00]
N40 G01 G90 X100 Y100 F0.1
N50 \#CS ON [0,0,0,0,0,45]
N60 G01 G90 X100 F0.1
N70 \#CS OFF
N80 \#TRAFO OFF
N90 Z[OSC OFF FEED=2.00]
N100 M30

```

\section*{21 Axis-specific programming}

A complete list of axis-specific additional functions is contained in the overview of commands in the Appendix under Additional axis-specific functions (<X>[..]) [> 816].

The programming syntax of the following NC commands is based on the axis-specific programming of positioning axes. After the axis identifier, parameterisation takes place by keywords and, if applicable, the associated values.

\subsection*{21.1 Selecting/deselecting axis compensations in the NC program (COMP)}

\section*{Release Note}

This function is available as of CNC Build V2.10.1501.00.

The various axis compensations [FCT-C5] can also be selected and deselected directly in the NC program in addition to the option of using the corresponding axis parameters. Axis-specific axis compensations for several axes in an NC block can be activated or deactivated simultaneously.

\section*{Notice}

Axis compensations switched off by the COMP command has a global NC program effect, i.e. compensations are not automatically activated at program end. They must be switched back on explicitly using the COMP command in the subsequent NC program.
```

<axis_name> [ COMP [ [ ON | OFF [ CROSS PLANE LEAD TEMP FRICT ]]|OFF_ALL ]
[NO_MOVE]{\}]

```
<axis_name>
COMP

ON
OFF
CROSS
PLANE
LEAD
TEMP
FRICT
CROSSTALK
OFF ALL

NO_MOVE

Name of the axis
Identifier to select/deselect axis-specific compensation. Must always be programmed as the first keyword.
Activates programmed compensation(s)
Deactivates programmed compensation(s)
Keyword for cross compensation
Keyword for plane compensation
Keyword for spindle leadscrew error compensation
Keyword for temperature compensation
Keyword for friction compensation [as of Build V2.11.2022.05]
Keyword for crosstalk compensation [as of Build V3.1.3079.32]
Switch off all active compensations. No further compensation keywords may be programmed after the keyword.
By default the position offset occurring when axis compensations are switched on/off is driven out before the NC program processing is continued. The keyword NO_MOVE suppresses this motion. The channel is initialised with the changed axis position. The position offset is only deactivated at the next axis motion programmed in the NC program.
Separator ("backslash") for clear programming of the command over multiple lines.

\section*{Programing Example}

\section*{Axis-specific programming}
```

(Deactivate cross and plane compensation in the X axis)
N10 X[COMP OFF CROSS PLANE
(Compensation programming of multiple axes in an NC block)
N50 X[COMP OFF CROSS] Y[COMP ON LEAD TEMP]
(Deactivate all compensations in the Z axis)
N100 Z[COMP OFF ALL]
(Deactivate all compensations of the Y axis without axis motion)
N200 Y[COMP OFF ALL NO MOVE]

```

\subsection*{21.2 Distance control (sensed spindles) (DIST_CTRL)}

\section*{Release Note}

The availability of this function depends on the configuration and the scope of the version.

If the tool-supporting axis (spindle axis) is equipped with the necessary hardware, this function specifies the distance of the tool to an uneven workpiece surface. The distance is detected by a measuring system and is continuously tracked by the NC on the uneven surface.

The distance control for a sensed spindle is enabled by the parameter P-AXIS-00328. It is activated by the following NC command. For more information please refer to the functional description "Distance control" [FCT-M3]
\begin{tabular}{|c|c|}
\hline <axis name> & [DIST_CTRL [ON | DRYRUN [ CONST_DIST ] | OFF | CHECK_POS | FREEZE | REF] SET_POS=.. SET_DIST=.. KP=.. I_TN=.. D_TV=.. FILTER_TYPE=. DISTC_N_CYCLES \(=\) =. DISTC_FG_FO=.. DISTC_ORDER=.. SMOOTH_FACT=.. KALMAN_SIGMA=.. [ NO_MOVE ] VAL1=.. - VAL5=.. \{ 1 \}] \\
\hline <axis name> & Name of the axis supporting the tool. \\
\hline DIST_CTRL & Identifier for the "Sensed spindles" function. Must always be programmed as first keyword. \\
\hline ON & Distance control on when workpiece surface is specified. A set position (SET_POS) must be set at switch-on. \\
\hline \begin{tabular}{l}
ON CONST_DIST \\
[as of V2.11.2804.03]
\end{tabular} & Distance control on when a constant distance to the workpiece surface is specified. A distance must be set with SET_DIST at switch-on. \\
\hline OFF & Distance control off. \\
\hline CHECK_POS & Check whether position is within the tolerance window. \\
\hline FREEZE & Freeze the control distance across the workpiece. The axis position or the output correction value is maintained. Axis tracking is interrupted. \\
\hline \begin{tabular}{l}
DRYRUN \\
[as of Build \\
V3.1.3079.23]
\end{tabular} & Distance control on when workpiece surface is specified. A set position (SET_POS) must be set at switch-on. If there are changes in the workpiece surface, the axis is not tracked. This permits data to be evaluated (e.g. filter effect) without controller feedback. \\
\hline \begin{tabular}{l}
DRYRUN CONST_DIST \\
[as of Build \\
V3.1.3079.23]
\end{tabular} & Distance control on when a constant distance to the workpiece surface is specified. A distance must be set with SET_DIST at switch-on. If there are changes in the workpiece surface, the axis is not tracked. This permits data to be evaluated (e.g. filter effect) without controller feedback. \\
\hline REF & Reference measuring system (sensor) (only if there is no absolute measuring system). \\
\hline SET_POS=<expr> & Specifying the workpiece surface in [ mm ] (absolute position). In the event of reset or program end, the set position is reset, i.e. a new set position must be specified before distance control is reactivated. \\
\hline SET_DIST=<expr> & Specifying a constant distance to the workpiece surface in [mm]. In the event of reset or program end, the distance is reset, i.e. a new distance must be specified before distance control is reactivated. \\
\hline \[
\begin{aligned}
& \text { KP=<expr> } \\
& \text { [as of Build } \\
& \text { V2.11.2809.06 }
\end{aligned}
\] & Weighting the distance control output values. Parameterisation can be executed analogous to P-AXIS-00759 . The value range is limited to \(0.0<\mathrm{kp}<=2.0\). For kp values less than 1.0, the distance control dynamics are reduced; for kp values greater than 1.0 the dynamics are increased. \\
\hline V3.1.3079.06] & A kp factor less than 1.0 reduces a possible distance control oscillation and steadies the control in the event of minor distance errors. \\
\hline \begin{tabular}{l}
|_TN=<expr> \\
[as of Build \\
V2.11.2809.06 \\
or \\
V3.1.3079.06]
\end{tabular} & Integral action time of the PID controller. The integral action time defines the time after which the P and I components of the manipulated variable are equal. Parameterisation can be executed analogous to P-AXIS-00764. The value range is limited to \(0.0<=\) I_TN \(<=50.0\). A large integral action time produces greater control stability. The shorter the integration action time, the greater the I component and the faster the control. A short integral action time excites oscillations more strongly. \\
\hline \begin{tabular}{l}
D_TV=<expr> \\
[as of Build \\
V2.11.2809.06 \\
or \\
V3.1.3079.06]
\end{tabular} & Derivative action time of the PID controller. The derivative action time defines the time after which the \(P\) and \(D\) components of the manipulated variable are equal. Parameterisation can be executed analogous to P-AXIS-00765. The value range is limited to 0.0 <= D_TV <= 2.0. The larger the derivative action time, the stronger the D component. \\
\hline FILTER_TYPE [as of Build V3.1.3079.23] & Filter type for filtering sensor values. \\
\hline DISTC N CYCLES [as of Build V3.1.3079.23] & Number of measurement values used for filtering. \\
\hline \[
\begin{aligned}
& \text { DISTC_FG_FO } \\
& \text { [as of Build }
\end{aligned}
\] & Limit frequency for low-pass filter [Hz]. \\
\hline
\end{tabular}

DISTC_ORDER
[as of Build
V3.1.3079.23]
SMOOTH_FACT [as of Build
V3.1.3079.23]
KALMAN_SIGMA [as of Build V3.1.3079.23]
NO_MOVE

VAL1=<expr> VAL5=<expr>

1

Order of low-pass filter.

Smoothing factor of the exponential average filter. Indicates the weighting of the current measurement value.

Uncertainty of measurement values recorded

By default, the resulting correction offset is deactivated when distance control is switched off. This motion can be suppressed by specifying NO_MOVE in combination with OFF. The channel is initialised with the changed axis position. The position offset is only deactivated at the next axis motion programmed in the NC program.
Freely assignable values (5)
Separator ("backslash") for clear programming of the command over multiple lines.
The keywords ON/OFF, FREEZE, CHECK_POS and REF in the same command sequence mutually cancel each other out.
The keywords SET_POS, SET_DIST and VAL1-VAL5 can always be programmed in combination with each other.

\section*{Notice}

If distance control is still active at program end, it is not automatically deselected.
When a reset or axis error occurs, active distance control is always deselected automatically.

\section*{Notice}

Parameters of the PID controller are not reset at program end.

\section*{Programing Example}

Programming examples for distance control
```

%DIST 1
; Set expected position of workpiece surface
N10 Z[DIST_CTRL SET_POS=30]
N20 Z[DIST_CTRL ON] ;Select
; ..
Nxx Z[DIST_CTRL OFF] ;Deselect
N999 M30

```
\%DIST 2
; Select + set expected position of workpiece surface
```

N10 Z[DIST_CTRL ON SET_POS=30]
; ...
Nxx Z[DIST_CTRL EREEZE] ;Hold position
; ...
Nxx Z[DIST CTRL OFF] ;Deselect
N999 M30

```

\section*{\%DIST_3}
; Select + set expected position of workpiece surface
N10 Z[DIST_CTRL ON SET_POS=50]
; Disable distance control, \(Z\) axis does not move
Nxx Z[DIST_CTRL OFF NO_MOVE]
; The generated compensation offset is considered for motion to the target position ; 100 .
Nxx G0 Z100
N999 M30

\section*{\%DIST_4}
; Set distance parameters
N10 Z[DIST CTRL SET POS=30 SET DIST=10]
; Select wíth workpiece surface specified (SET_POS)
N20 Z[DIST_CTRL ON]
; ...
Nxx Z[DIST_CTRL OFF] ; Deselect
; ...
; Select with workpiece surface specified (SET_DIST)
NXx Z[DIST_CTRL ON CONST_DIST]
; ...
Nxx Z[DIST_CTRL OFF ;Deselect
N999 M30

\section*{\%DIST 5}

N10 Z[DIST_CTRL FILTER_TYPE=KALMAN_MA] ; Select filter type
; Parameterise the filter
N20 Z[DIST_CTRL DISTC_N_CYCLES=30 KALMAN_SIGMA=1000]
; Enable distance control
N30 Z[DIST_CTRL ON CONST_DIST SET_DIST=1].
; ...
; Change filter
NXX Z[DIST_CTRL FILTER_TYPE=KALMAN_EXPO SMOOTH_FACT=0.3]

Nxx Z[DIST CTRL OFF]; Deselect
N999 M30

\subsection*{21.3 Programmable axis override (OVERRIDE)}

This command allows for the axis feed, if required the different influencing of feed and rapid feed blocks in the NC program. The axis-specific programmed override is active during path motions if the axis is moving. This does not affect the mode of operation of real-time influencing of feed by the PLC.

In addition a programmable path override [ 460] function is also provided.
When several axes are moved in the same NC block with different axis-specific override values, the smallest override always takes effect. If an additional path override is also defined, the effective override results from multiplying the two override values.

\section*{Notice}

The G166 [» 185] function suppresses the programmed override values.
```

<axis_name> [ OVERRIDE FEED_FACT<expr> RAPID_FACT<expr> { \ } ]

```

Name of the axis
Identifier for axis-specific override programming. Must always be programmed as first keyword.
Override factor for feed blocks in [0.1\%-200\%]
Override factor for rapid traverse blocks [0.1\%-200\%]
Separator ("backslash") for clear programming of the command over multiple lines.

\section*{Programing Example}

\section*{Programmable axis override}
```

%ax_override
N10 G01 X100 Y100 Z100 F1000
N40 X[OVERRIDE FEED FACT=20 RAPID FACT=60] Axis override X G01 20%, G00 60%
N50 Y[OVERRIDE FEED FACT=30 RAPID FACT=70] Axis override Y G01 30%, G00 70%
N60 Z[OVERRIDE FEED_FACT=40 RAPID_FACT=80] Axis override Z G01 40%, G00 80%
N50 G00 X0 G00 motion with 60% override
N60 Y0 G00 motion with 70% override
N70 Z0 G00 motion with 80% override
N80 G01 X100 F2000 G01 motion with 20% override
N90 Y100 G01 motion with 30% override
N100 Z100 G01 motion with 40% override
N110 X200 Y200 G01 motion with 20% override
N120 X300 Y300 Z200 G01 motion with 20% override
M30

```

\subsection*{21.4 Programmable acceleration overload (DYNAMIC)}

For technological reasons it may be necessary to exceed the specified dynamic limit values of the drive in connection with contour affecting influencing processes e.g. to ensure a constant path velocity on polynomial contours.
The following axis-specific command combined with the assigned parameter P-AXIS-00394 permits a weighting of the axis dynamic in percent \(\%\) in excess of the permissible maximum acceleration P-AXIS-00008. P-AXIS-00394 represents the permissible upper limit for the acceleration weighting factor of the axis in per mill \%. The weighting factor refers to the feed dynamic limit values of the corresponding active slope profile.

Currently the acceleration weighting function can be used in conjunction with Contouring mode 6.
```

<axis_name> [ DYNAMIC DIST_SOFT | ACC_FACT<expr> { \ } ]

```
```

<axis_name>
DYNAMIC
DIST_SOFT
ACC_FACT<expr>
l
Name of the axis
Identifier for axis dynamic weighting. Must always be programmed as the first keyword.
Identifier for the polynomial contouring mode 6
Axis-specific weighting factor in \%
Separator ("backslash") for clear programming of the command over multiple lines.

```

\section*{Notice}

The minimum weighting value is \(100 \%\).
The maximum weighting value is limited to P-AXIS-00394.

\section*{Programing Example}

\section*{Programmable acceleration overload}
```

%dynamic
N10 \#SLOPE[TYPE=STEP]
N20 \#CONTOUR MODE[DIST_SOFT PATH_DIST=35 ACC_MAX=100 ]
N30 C[DYNAMIC DIST_SOFT ACC_FACT=200]
(* Acceleration overerload factor for C axis 200% *)
N30 G1 G91 G261
N40 X59.485 F10000
N50 X105.172 C26.992
N60 X113.189 C46.171
N70 X100.348 C-46.171
N80 X99.179 C-26.992
N90 G260 X138.799
N100 G261
M30

```

\section*{21.5 \\ Synchronising an axis in coordinated motion (SYNC IN / OUT)}

\section*{Release Note}

This function is available as of CNC Build V2.11.2013.22

Some specific processes require a synchronised motion of a single axis (slave axis) in combination with a coordinated motion. At certain programmed positions it is required that the slave axis is located at a specific position and moves at a specific velocity. The slave axis then moves at the synchronised velocity until synchronisation is cancelled.
Typical application examples include machines with the continuous throughput of endless material. The material must then be cut at a specific place during the coordinated motion. At a specific master position (workpiece length) the rotating knife must be placed in cutting position. The knife then moves at constant velocity until the cut is finished.


Fig. 191: Synchronised cutting

\section*{Restrictions:}

An axis cannot be synchronized if:
- the axis is moving in coordinated motion at the time.

\section*{Configuration:}

To use this function, the following setting must be made in the start-up list ([STUP]): configuration.channel[0].path_preparation.function FCT_DEFAULT \| FCT_SYNC configuration.channel[0].interpolator.function FCT_IPO_DEFAULT | FCT_SYNC

\section*{Syntax for programming synchronous motions:}
\begin{tabular}{|c|c|}
\hline \begin{tabular}{l}
<axis_name> [ SYNC IN| \\
FEED_MAX
\end{tabular} & \begin{tabular}{l}
G90 | G91 G00 | G01 FEED<expr> \\
GHT<expr> POS<expr> DIST<expr> \{ \(\backslash\}]\)
\end{tabular} \\
\hline <axis_name> & Name of the axis to be synchronised \\
\hline SYNC & Identifier for synchronised axis motion. Must always be programmed as the first keyword. \\
\hline IN & Identifier to mark the start of synchronised motion. \\
\hline OUT & Identification to mark the end of synchronised motion. \\
\hline G90 / G91 & Absolute/relative dimension \\
\hline G00 / G01 & Rapid traverse/linear interpolation \\
\hline FEED<expr> & Axis-specific feedrate in [mm/min, m/min, inch/min] \\
\hline FEED_MAX_WEIGHT<expr> & Weighting factor in [\%], referred to axis-specific maximum feed P-AXIS-00212. Only weighting values less than \(100 \%\) are permitted (according to G194 [ 157]). \\
\hline DIST<expr> & Axis position in [mm, inch] at which the synchronous velocity is reached. \\
\hline 1 & Distance in [mm, inch] at which the synchronous velocity is moved. \\
\hline & Separator ("backslash") for clear programming of the command over multiple lines. \\
\hline
\end{tabular}

\section*{Programing Example}

\section*{Synchronising an axis in coordinated motion}
```

%sync
NO10 G90 X0 Y0 Z0 A0
N020 G91 F5000
NO30 X=67.913 A[SYNC IN G01 FEED MAX WEIGHT=100 G91 POS=130 DIST=70]
;A axis reaches maximum velocity on axis position 130,
;while X axis is reaching position 67.913 at this point
N040 X=1.5 ; A and X axes move synchronously, in doing so the velocity
;of the X axis is then defined so that the X axis covers a distance of
3 mm
N050 X=1.5 ; while the A axis moves 70' at maximum velocity
NO60 X=14.541 A[SYNC OUT G91 GO POS160] G261
;At beginning of this block synchronous motion is cancelled
Path axes move again at the programmed feed rate; the A axis
moves independently to the specified position
N070 X=15.862 Z=1.248 Y=0.185
N080 X=15.992 Z=1.889 Y=0.213
N090 X=32.243 Z=3.306 Y=0.482
N100 X=22.186
N110 X=31.696 Z=-2.597 Y=-0.389
N120 X=25.297 Z=-3.846 Y=-0.491
N130 X=39.819 A[SYNC IN G01 FEED_MAX_WEIGHT=100 G91 POS=130 DIST=70]
N140 X=1.257
N150 X=1.257
N160 X=200 A[SYNC OUT G91 GO POS160]
N180 M30

```
kernel Industrielle Steuerungstechnik GmbH

\subsection*{21.6 Programming an axis polynomial (POLY)}

Release Note
This function is available as of CNC Build V2.11.2016.08

\section*{Characteristics of axis-specific polynomials}

The motion rules for an axis can be programmed by specifying axis-specific polynomials.
This axis-specific polynomial motion is programmable for linear motions (G00, G01). The dynamic parameters of the currently active G function (G00 or G01) are used.

An upper limit of the polynomial parameter is specified for each axis polynomial up to which the polynomial parameter is interpolated. If the upper limit is not programmed, the value 1.0 is assigned.

The polynomial coefficients of an axis polynomial are defined in square brackets after the axis identifier in ascending order. The keyword POLY must always be specified first for identification purposes. Higher polynomial coefficients which are not required can be omitted. Coefficients which are not programmed are assigned the value 0 . At least the first coefficient 'A0' must be set.

The maximum possible degree of the polynomial is 5 .

\section*{Evaluation}

The polynomial coefficients refer to the specification of absolute axis positions in a 5th degree polynomial:
\(p(s)=A 0+A 1^{*} s+A 2{ }^{*} s^{2}+A 3^{*} s^{3}+A 4^{*} s^{4}+A 5 s^{5}\)
The polynomial parameter is interpolated for the polynomial from zero up to the programmed upper limit simultaneously to the executed motion path.

This applies to the absolute position of the polynomial axis in [mm or \({ }^{\circ}\) ]:
At motion start \((s=0)\) :
\[
p(0)=A 0
\]

At motion end \((s=L)\) :
\[
p(L)=A 0+A 1^{*} L+A 2^{*} L^{2}+A 3^{*} L^{3}+A 4^{*} L^{4}+A 5 L^{5}
\]

Axis-specific polynomial programming is only effective in the current NC block (non-modal). Therefore, if required, it must be reprogrammed in the next motion block for each axis.

Notice
When determining the polynomial coefficients, it is important to ensure that the axis position is continuous, i.e. the polynomial value at zero position must correspond to the axis position of the previous motion block.
The repeated programming of axis polynomials in sequenced NC blocks requires that the end position of a polynomial corresponds to the start position of the next polynomial.
Since the value at position 0 is only defined by the coefficient A 0 for a polynomial, the following applies: A 0 is equal to the axis position from the previous motion block.

\section*{Programming}

\section*{Scheme}

Axis [ POLY L<Maximum value of polynomial parameter> A0 A1 A2 A3 A4 A5 ]

\section*{Example:}

X [ POLY L=1.0 A0=0.1 A1=0.2 A2=0.3 A3=0.4 A4=0.5 A5=0.6 ]
The same NC block may contain a mixed programming of linear motions and one* axis-specific polynomial. Active offsets (G54, G92, \#PSET...) may be included in the programmed polynomial positions.

No calculation or monitoring of the dynamic is executed for the programmed polynomial axis. Similarly, no command value monitoring of software limits takes place (only actual value specific monitoring of limits).

Syntax for programming axis-specific polynomials:
\begin{tabular}{|c|c|}
\hline <axis_name> & Name of the polynomial axis \\
\hline POLY & Identifier for the polynomial programming of an axis. Must always be programmed as the first keyword. \\
\hline L<max. poly param> & Upper limit of the polynomial parameter of the axis to be moved without unit (optional: if not programmed, \(L\) has value 1.0 , programmed value must be \(>0\) ) \\
\hline A0<a0> & First polynomial coefficient, mandatory (start value of polynomial) \\
\hline A1<a1>-A5<a5> & Second to sixth polynomial coefficient, \(x, x^{\wedge} 2, x^{\wedge} 3, x^{\wedge} 4, x^{\wedge} 5\) (optional: default value of non-programmed coefficients is 0 ) \\
\hline
\end{tabular}

\section*{Programing Example}

Programming an axis polynomial
```

;C axis, polynomial parameter L = 0.7 A0 = 0.1, A1 = 0.3, A2 = 0.5
Nxx C0.1
Nxx C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]
;X axis, polynomial parameter L = 0.3, A0 = 0.2, A1= 0.5
NxX X0.2
Nxx X[POLY L=0.3 A0=0.2 Al=0.5]
; Simple programming without polynomial parameter (default L1), only A0 coefficient
Nxx X0.2
Nxx X[POLY A0=0.2]
;Mixed programming of linear motion and axis polynomial
Nxx C0.1
Nxx G01 F1000 X100 Y150 C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]
;Note: The equals signs between the keyword and the value are optional

```

\subsection*{21.7 Set an axis position in the channel}

Release Note
This function is available as of CNC Build V2.11.2808

This command sets the current position of an axis in the NC channel to a defined value. This value acts on the ACS level (in the position controller). It does not initiate a motion. Instead, it marks the axis as referenced after the axis position is repositioned. The NC channel is then initialised with the new axis positions, taking into consideration any active offsets.
The new axis position is specified with as an absolute value (POS) or a relative value to the current position (OFFSET).
```

<axis_name> [SET_POSITION POS=.. | OFFSET=.. {\} ]

```
\begin{tabular}{ll} 
<axis_name> & \begin{tabular}{l} 
Name of the axis \\
SET_POSITION
\end{tabular} \\
\begin{tabular}{l} 
Identifier for the function of setting an axis position. Must always be programmed as first \\
keyword.
\end{tabular} \\
POS=<expr> & New defined absolute axis position in [mm, inch] \\
OFFSET=<expr> & \begin{tabular}{l} 
Relative offset to the current axis position in [mm, inch] \\
Separator ("backslash") for clear programming of the command over multiple lines.
\end{tabular} \\
\hline
\end{tabular}

Programing Example
Set an axis position
```

%set_pos.nc
N010 G01 F2000 X0 Y0 Z0 A0 B0 C0
N020 \$FOR P1=0,100,1
N030 G91 X100 ;Axis X moves to 10000mm
N040 \$ENDFOR
N050 X[SET_POSITION POS=100] ; Set X axis position to 100
; ..
; . .
N999 M30

```

\subsection*{21.8 Lifting/lowering an axis (LIFT)}

For more information please refer to the functional description "Collision detection by lift function" [FCT-A11].

\section*{Cross-block lifting/lowering}

Programming is based on the syntax for independent axes. The corresponding parameters can be programmed at the start of lifting/lowering. These are non-modal parameters, i.e. if required they are reset for every start.
```

<axis_name> [ LIFT_START [ DOWN ] [ G90 | G91 ] [ POS<expr> ] POS_LIMIT<expr> ]

```
<axis_name>
LIFT_START
DOWN

G90 / G91 Absolute/relative dimension; the default dimension is G90. G91 is non-modal and is
Lift axis name
Identifier for the start of the (cross-block) independent lifting motion of the axis.
The axis motion direction can be inverted via DOWN, i.e. the motion is in the direction of the negative software limit switch. If nothing is specified, the default direction is in the direction of the positive software limit switch. only active for the lifting/lowering motion.
POS<expr> Target position of the lift axis after the lifting motion. The current command position of the axis (see V.A.ABS.<axis name>) is the default.
POS_LIMIT<expr> Maximum lifting height or lowering depth
```

<axis_name> [ LIFT_END ]

```
\begin{tabular}{ll} 
<axis_name> & Lift axis name \\
LIFT_END & Identifier for the end of the (cross-block) independent lifting motion of the axis.
\end{tabular}

\section*{Programing Example}

Cross-block lifting/lowering
```

N10 X10 Y20 Z30 ;Cut with laser
N20 M5 ;Laser off
N30 Z[LIFT_START POS=12 POS_LIMIT=100] ;Lift Z axis
N30 G01 X.. Y..
N40 G02 X.. Y..
N50 G03 X.. Y..
N60 G01 X.. Y..
N70 Z[LIFT_END] ;Absolutely lower Z axis to target 12 mm
N80 M4 ;Laser on
N90 X20 Y20 ...
N10 X10 Y20 Z30
N30 Z[LIFT_START POS=12 POS_LIMIT=100] ;Lift Z axis
N40 G01 X.. Y..
N50 G01 X.. Y..
N60 Z[LIFT_END] ;Absolutely lower Z axis to target 12 mm
N70 X100

```
alternative programming
N110 X10 Y20 Z30
N140 G01 X.. Y.. Z[LIFT_START POS=12 POS_LIMIT=100]
N150 G01 X.. Y.. Z[LIFTEND]
N170 X100

\section*{Lifting/lowering in an NC block}

Programming is based on the syntax for independent axes. The corresponding parameters can be programmed at the start of lifting/lowering. These are non-modal parameters, i.e. if required they are reset for every start.
```

<axis_name> [ LIFT [ DOWN ] [G90 | G91] [POS<expr>] POS_LIMIT<expr> ]

```
<axis_name>
LIFT

DOWN

G90 / G91

POS<expr>

POS_LIMIT<expr>

Lift axis name
Identifier for the start and end of the independent lifting motion of the axis in the current NC block
The axis motion direction can be inverted via DOWN, i.e. the motion is in the direction of the negative software limit switch. If nothing is specified, the default direction is in the direction of the positive software limit switch (option not available as at 10/2011).
Absolute/relative dimension. The default dimension is G90. G91 is non-modal and is only active for the lifting/lowering motion.
Target position of the lift axis after the lifting motion. The current command position of the axis (see V.A.ABS.<axis name>) is the default.
Maximum lifting height or lowering depth

\section*{Programing Example}

\section*{Lifting/lowering in an NC block}
```

; single-row programming
N200 z40
N240 X10 Y.. Z[LIFT POS=30 POS_LIMIT=300]
N250 X20 Y.. Z[LIFT POS=20 POS LIMIT=300]
N260 X30 Y.. Z[LIFT POS=25 POS LIMIT=300]
N270 X.. Y.. Z[LIFT POS=30 POS_LIMIT=300]
N280 X.. Y.. Z[LIFT POS=30 POS_LIMIT=300]

```


Fig. 192: Single-row lifting

\subsection*{21.9 Moving to fixed stop}

For more information please refer to the functional description "Moving to fixed stop" [FCT-M8]
```

<Achsname> [ FIXED_STOP [ ON | OFF ] [ TORQUE_LIMIT=..] [ POS_LAG_LIMIT=..]
[ CYCLES=..] [ WINDOW=..] [ START=..] [ END=..]
[ERR_NOT_DETECTED=..] {\}]

```
\begin{tabular}{ll} 
<Achsname> & \begin{tabular}{l} 
Name of the axis to be used with the "Move to fixed stop" function. \\
Activate the "Move to fixed stop" function for this axis. \\
ON \\
Motion information must also be specified for the axis.
\end{tabular} \\
OFF & \begin{tabular}{l} 
Deactivate the "Move to fixed stop" function. \\
In addition a motion should be programmed for the axis to move away from the \\
fixed stop.
\end{tabular} \\
Specifying the torque limit with "Move to fixed stop". \\
SORQUE_LIMIT=<expr> \\
Scaling is determined by parameterising the "Move to fixed stop" function from the \\
axis parameters (see P-AXIS-00724). Normally this is given in percent (\%) of the \\
drive nominal torque. \\
If no torque is specified for the axis in the NC program, the default value in the axis \\
parameter P-AXIS-00729 is used for the torque limit.
\end{tabular}

\section*{22 Appendix}

\subsection*{22.1 Overview of commands}
22.1.1 G functions (G..)
\begin{tabular}{|c|c|c|}
\hline G00 & Linear interpolation in rapid traverse & p. [> 53] \\
\hline G01 & Linear interpolation & p. [> 54] \\
\hline G02 & Clockwise circle (cw) & p [ 55] \\
\hline G03 & Counter-clockwise circle (ccw) & p. [> 55] \\
\hline G02 Z.. K.. & Helical interpolation (cw) & p. [> 59] \\
\hline G03 Z.. K.. & Helical interpolation (ccw) & p. [> 59] \\
\hline G04 & Dwell time & p. [> 83] \\
\hline G05 & Tangential selection/deselection of TRC & p. [ 556] \\
\hline G08 & Acceleration at block start & p. [ 109] \\
\hline G09 & Deceleration at block end & p. [ 109] \\
\hline G10 & Feedrate of TRC, constant & p. [ 559] \\
\hline G11 & Feedrate of TRC, adapted & p. [ 559] \\
\hline G12 & Deselecting corner deceleration & p. [ 131] \\
\hline G13 & Selecting corner deceleration & p. [ 131] \\
\hline G17 & X-Y plane & p. [ 114] \\
\hline G18 & Z-X plane & p. [ 114] \\
\hline G19 & Y-Z plane & p. [ 114] \\
\hline G20 & Deselect the mirroring function & p. [ 115] \\
\hline G21 & Mirror programmed paths on the Y axis & p. [> 115] \\
\hline G22 & Mirror programmed paths on the X axis & p. [> 115] \\
\hline G23 & Superimpose G21 and G22 & p. [ 115] \\
\hline G25 & Linear transitions with TRC & p. [ 498] \\
\hline G26 & Circular transitions with TRC & p. [ 498] \\
\hline G33 & Thread cutting, uniform pitch & p. [ 642] \\
\hline G40 & Deselection of TRC/SRK & \\
\hline & - Tool radius compensation (TRC) & p. [ 498\(]\) \\
\hline & - Cutter radius compensation (G40/G41/G42) & p. [ 635] \\
\hline G41 & TRC/SRK left of contour & \\
\hline & - Tool radius compensation (TRC) & p. [ 498\(]\) \\
\hline & - Cutter radius compensation (G40/G41/G42) & p. [ 635] \\
\hline G42 & TRC/SRK right of contour & \\
\hline & - Tool radius compensation (TRC) & p. [> 498] \\
\hline & - Cutter radius compensation (G40/G41/G42) & p. [> 635] \\
\hline G51 & Selection of diameter programming & p. [ 633] \\
\hline G52 & Deselection of diameter programming & p. [ 633] \\
\hline G53 & Deselect zero offsets & p. [ 132] \\
\hline G54-G59 & Select zero offsets & p. [ 132] \\
\hline G60 & Exact stop & p. [ 126] \\
\hline G61 & Polynomial contouring & p. [ 127] \\
\hline G63 & Thread tapping & p. [ 646] \\
\hline G66 & Synchronise cycle at block end & p. [ 189] \\
\hline G68 & Select contour rotation & p. [ 190] \\
\hline G69 & Deselect contour rotation & p. [ 190] \\
\hline G70 & Inputs in inch (inch) & p. [ 123] \\
\hline G71 & Inputs in metric units & p. [ 123] \\
\hline \multirow[t]{4}{*}{G74} & Homing & \\
\hline & - Programmable homing & p. [> 84] \\
\hline & - Homing in DIN syntax & p. [ 667] \\
\hline & - Homing in spindle-specific syntax & p. [ 675] \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline \multicolumn{2}{|l|}{kernel Industrielle Steuerungstechnik GmbH} & Appendix \\
\hline G80-G89 & Implicit subroutine calls & p. [> 123] \\
\hline G90 & Dimension systems, absolute dimension & p. [ 124] \\
\hline G91 & Dimension systems, incremental dimension (all parameters relative) & p. [> 124] \\
\hline G92 & Reference point offset & p. [ 85] \\
\hline G93 & Switching F word to machining time in seconds & p. [ 157] \\
\hline G94 & Switching F word to feedrate per minute & p. [> 157] \\
\hline \multirow[t]{3}{*}{G95} & Feedrate per revolution & \\
\hline & - Switching F word to feedrate per revolution & p. [> 157] \\
\hline & - Feedrate per revolution for turning & p. [> 637] \\
\hline G96 & Switching S word to constant cutting velocity & p. [> 639] \\
\hline G97 & Switching S word to spindle speed & p. [> 639] \\
\hline G98 & Setting negative software limit switch & p. [> 86] \\
\hline G99 & Setting positive software limit switch & p. [> 88] \\
\hline \multirow[t]{5}{*}{G100} & Measuring functions & p. [> 90] \\
\hline & Measuring with multiple axes (Type 1) & p. [> 92] \\
\hline & Measuring with a single axis (Type 2) & p. [> 94] \\
\hline & Measuring with main axes (Type 4) & p. [> 98] \\
\hline & Measuring with motion to a fixed stop (Type 7) & p. [ 101] \\
\hline G101 & Include measuring offset calculation in offset & p. [ 102] \\
\hline G102 & Extract measuring offset from offset & p. [ 102] \\
\hline G106 & Measurement run with motion to target point (Type 3) & p. [> 96] \\
\hline G107 & Deselect cross-block edge banding & p. [> 107] \\
\hline G108 & Edge banding & p. [> 104] \\
\hline G112 & Gear change & p. [> 184] \\
\hline G115 & Disabling look-ahead function & p. [> 185] \\
\hline G116 & Disabling the calculation of block transition velocity & p. [ 185] \\
\hline G117 & Enabling the complete look-ahead function & p. [ 185] \\
\hline G127 & Weighting of maximum velocity, axis-specific & p. [ 149] \\
\hline G128 & Weighting of maximum velocity, axis group-specific & p. [> 149] \\
\hline G129 & Weighting of rapid traverse velocity & p. [ 149] \\
\hline G130 & Axis-specific weighting of acceleration & p. [ 150] \\
\hline G131 & Axis group-specific weighting of acceleration for G01/G02/G03 & p. [ 150] \\
\hline G132 & Axis group-specific weighting of ramp time & p. [> 153] \\
\hline G133 & Axis group-specific weighting of ramp time for G01/G02/G03 & p. [> 153] \\
\hline G134 & Axis group-specific weighting of geometrical ramp time & p. [> 153] \\
\hline \multirow[t]{3}{*}{G135} & Selecting feedforward control & \\
\hline & - Feedforward control (G135/G136/G137) & p. [ 148\(]\) \\
\hline & - Spindle-specific feedforward control (G135, G136, G137) & p. [ 679] \\
\hline \multirow[t]{3}{*}{G136} & Specifying weighting of feedforward control & \\
\hline & - Feedforward control (G135/G136/G137) & p. [> 148] \\
\hline & - Spindle-specific feedforward control (G135, G136, G137) & p. [ 679] \\
\hline \multirow[t]{3}{*}{G137} & Deselecting feedforward control & \\
\hline & - Feedforward control (G135/G136/G137) & p. [ 148] \\
\hline & - Spindle-specific feedforward control (G135, G136, G137) & p. [ 679] \\
\hline \multirow[t]{3}{*}{G138} & Direct TRC selection/deselection & \\
\hline & - Direct selection & p. [ 507] \\
\hline & - Direct deselection & p. [ 516] \\
\hline \multirow[t]{2}{*}{G139} & Indirect TRC selection/deselection & \\
\hline & - Indirect selection with G25 & p. [> 510] \\
\hline
\end{tabular}
kernel Industrielle Steuerungstechnik GmbH
\begin{tabular}{|c|c|c|}
\hline & - Indirect selection with G26 & p. [> 513] \\
\hline & - Indirect deselection with G25 & p. [> 519] \\
\hline & - Indirect deselection with G26 & p. [ 522] \\
\hline G140 & Deselecting contour masking & p. [> 561] \\
\hline G141 & Selecting contour masking & p. [> 561] \\
\hline G150 & Deselecting spline interpolation & \\
\hline & - Deselecting Akima spline interpolation & p [> 292] \\
\hline & - Deselecting B spline interpolation & p [> 298] \\
\hline G151 & Selecting spline interpolation & p. [> 291] \\
\hline G159 & Extended zero offsets & p. [> 137] \\
\hline G160 & Enabling/disabling zero offsets axis-specific & p. [ 138] \\
\hline G161 & Specifying centre point for circle definition, absolute & p. [ 139] \\
\hline G162 & Specifying centre point for circle definition, relative (initial state) & p. [> 139] \\
\hline G163 & Selecting radius programming & p. [> 140] \\
\hline G164 & Deselecting circle centre point correction & p. [> 144] \\
\hline G165 & Selecting circle centre point correction & p. [> 144] \\
\hline G166 & Override 100\% & p. [> 188] \\
\hline G167 & Spindle override 100\% & \\
\hline & - Homing in DIN syntax & p. [> 668] \\
\hline & - Override in spindle-specific syntax & p. [ 676] \\
\hline G193 & Path-related feed interpolation & p. [> 112] \\
\hline G194 & Switching F word to weighting of maximum feedrate & p. [ 157] \\
\hline G196 & Maximum spindle speed for G96 & p [ 639] \\
\hline G200 & Selecting manual mode without parallel interpolation & p. [> 171] \\
\hline G201 & Selecting manual mode with parallel interpolation & p. [> 169] \\
\hline G202 & Deselecting manual mode with parallel interpolation & p. [> 169] \\
\hline G231 & Axis group-specific weighting of acceleration for G00 & p. [> 150] \\
\hline G233 & Axis group-specific weighting of ramp time for G00 & p. [> 153] \\
\hline G236 & Direct TRC selection/deselection on the path & \\
\hline & - Direct TRC selection/deselection on the path & p. [ \({ }^{\text {[ }}\) 540] \\
\hline & - Selecting/deselecting with closed contours & p. [ 544] \\
\hline G237 & Perpendicular TRC selection/deselection & p. [ 525] \\
\hline G238 & Selecting inside corner of TRC & p. [> 532] \\
\hline G239 & Selecting/deselecting TRC directly without block & p. [ 535] \\
\hline G260 & Deselecting polynomial contouring & p. [> 127] \\
\hline G261 & Selecting polynomial contouring (at block end) & p. [> 127] \\
\hline G293 & Time-related feed interpolation & p. [> 112] \\
\hline G301 & Inserting chamfers & p. [> 158] \\
\hline G302 & Inserting roundings & p. [> 158] \\
\hline G303 & Arc in 3D space & p. [ 66] \\
\hline G310 & Measuring with interruption and jump (G310) (Types 5, 6) & p. [> 100] \\
\hline G331 & Tapping with pitch specification & p. [ 648] \\
\hline G332 & Tapping retraction with pitch specification & p. [> 648] \\
\hline G333 & Acceleration weighting with feedhold, axis-specific & p. [> 150] \\
\hline G334 & Acceleration weighting with feedhold, axis group-specific & p. [ 150] \\
\hline G338 & Ramp time weighting with feedhold, axis-specific & p. [ 153] \\
\hline G339 & Ramp time weighting with feedhold, axis group-specific & p. [> 153] \\
\hline G351 & Mirroring with axis specification & p. [> 120] \\
\hline G359 & Deselecting exact stop & p. [> 126] \\
\hline
\end{tabular}
\begin{tabular}{lll} 
G360 & Selecting exact stop & p. [ 126] \\
G800 - G819 & Implicit subroutine calls (additional) & p. [ 123] \\
G900 & Deceleration at block end & p. [ 109] \\
G901 & Deceleration at block end & p. [ 109]
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline 22.1.2 & M functions (M..) & \\
\hline мо0 & Programmed stop & p [ 194] \\
\hline M01 & Optional stop & p [ 194] \\
\hline M02 & Program end & p [ 194] \\
\hline M03 & Spindle rotation cw & \\
\hline & - in DIN syntax & p. [ 624] \\
\hline & - in spindle-specific syntax & p. [ 670] \\
\hline M04 & Spindle rotation ccw & \\
\hline & - in DIN syntax & p. [ \({ }^{\text {c 624] }}\) \\
\hline & - in spindle-specific syntax & p. [ 870\(]\) \\
\hline M05 & Stop spindle & \\
\hline & - in DIN syntax & p. [ \({ }^{\text {P 624] }}\) \\
\hline & - in spindle-specific syntax & p. [ 670] \\
\hline M06 & Calling a tool change program & p [ 195] \\
\hline M17 & Subroutine end & p [ 194] \\
\hline M19 & Positioning spindle & \\
\hline & - in DIN syntax & p. [ 625] \\
\hline & - in spindle-specific syntax & p. [ \({ }^{\text {b }}\) 672] \\
\hline M29 & Subroutine end & p [ 194] \\
\hline M30 & Program end & p [ 194] \\
\hline M40-45 & Selecting spindle gear stages & p [ 630] \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline 22.1.4 & Control block statements (\$..) & \\
\hline \$BREAK & BREAK statement & S. [> 245] \\
\hline \$CONTINUE & CONTINUE statement & S. [> 246] \\
\hline \$DO & DO loop & S. [> 243] \\
\hline \multicolumn{3}{|l|}{\$ENDDO} \\
\hline \$REPEAT & REPEAT loop & S. [> 243] \\
\hline \multicolumn{3}{|l|}{\$UNTIL} \\
\hline \$FOR & FOR loop & S. [> 241] \\
\hline \multicolumn{3}{|l|}{\$ENDFOR} \\
\hline \$GOTO & GOTO statement & S. [> 236] \\
\hline \$IF & & S. [> 232] \\
\hline \$ELSE & The IF-ELSE branching & \\
\hline \multicolumn{3}{|l|}{\$ELSEIF} \\
\hline \multicolumn{3}{|l|}{\$ENDIF} \\
\hline \$SWITCH & & S. [> 236] \\
\hline \$CASE & Switch branching & \\
\hline \multicolumn{3}{|l|}{\$DEFAULT} \\
\hline \multicolumn{3}{|l|}{\$ENDSWITCH} \\
\hline \$WHILE & WHILE loop & S. [> 243] \\
\hline \$ENDWHILE & & \\
\hline
\end{tabular}

\subsection*{22.1.5 Additional functions (\#..)}

\section*{A}
\begin{tabular}{|c|c|c|}
\hline \#ACS ON/OFF & Defining/activating a clamping position compensation coordinate system & p. [> 732] \\
\hline \#ADD & Additional information at block end & p. [> 370] \\
\hline \#AKIMA STARTVECTOR & Defining start tangent & p. [> 294] \\
\hline \#AKIMA ENDVECTOR & Defining end tangent & p. [> 294] \\
\hline \#AKIMA TRANS & Specifying spline curve motion block & p. [> 293] \\
\hline \#ANG & Contour line programming & p. [> 68] \\
\hline \#AX DEF & Defining an axis configuration (extended syntax) & p. [> 333] \\
\hline \#AX DEF DEFAULT & Loading default axis configuration (extended syntax) & p. [> 333] \\
\hline \#AX LINK ON/OFF/OFF ALL & Programming axis couplings & \\
\hline & - extended syntax & p. [> 355] \\
\hline & - Extension "SOFT-GANTRY" & p. [ 357] \\
\hline \#AX LOCK/UNLOCK ALL & Locking an axis motion during PTP & p. [ 722] \\
\hline \#AX REQUEST & Requesting axes (extended syntax) & p. [> 322] \\
\hline \#AX RELEASE & Releasing axes (extended syntax) & p. [> 330] \\
\hline \#AX RELEASE ALL & Releasing all axes (extended syntax) & p. [> 330] \\
\hline
\end{tabular}

B
\#BACKWARD STORAGE CLEAR Clearing backward storage
p. [ 456]
\#BCS DEF
\#BCS ON/OFF
\#BLOCKSEARCH LOCKED/
Defining and storing a BCS
p. [ 739]

Defining, storing and activating a BCS
p. [ 739]

Lock program areas for block search
kernel Industrielle Steuerungstechnik GmbH
```

C

```
\#CACHE LOAD/CLEAR/ALL
\#CALL AX
\#CAX
\#CAX OFF
\#CAXTRACK ON/OFF
\#CHANNEL INIT

Load NC programs to local cache
Requesting axes
Requesting a spindle axis for C axis machining
Returning C axis to the spindle
Automatic axis tracking
Initialising channel with current command/actual positions
- Initialising channel with command positions
p. [> 179]
- Initialising channel with actual positions
p. [ 181]
[FCT-C30]
Setting function-specific parameters in the channel
- Feed programming for microjoints
- Time offset with estimation
[FCT-C1]
[FCT-C34]
Inserting chamfers and roundings: chamfer length p. [> 158]
Inserting chamfers and roundings: chamfer width p. [ 158]
Deleting a saved configuration
p [ 310]
Non-synchronised write of SERCOS commands p. [> 378]
Synchronised write of SERCOS commands
Non-synchronised wait for SERCOS commands
p. [> 379]
p. [ 380]

Synchronised wait for SERCOS commands
p. [ 381]

Cross-block comments
p. [ 343]

Contouring
- Smoothing methods p. [ 247]
- Parameterise polynomial contouring p. [> 268]

Defining a control area
Selecting/deselecting control area
p. [ 446]

Clearing a control area
Parameterising corner deceleration
Defining and storing a CS
Defining, storing and activating a CS
Changing the rotation sequence of CS axes
p. [ 449]
\#CONTROL AREA ON/OFF
\#CONTROL AREA CLEAR
\#CORNER PARAM
\#CS DEF
\#CS ON/OFF
\#CS MODE ON/OFF
\#CYL
\#CYL OFF

\section*{\#CYL ORI LATERAL \\ \#CYL ORI PROFILE}

Lateral surface machining
- Selection
p. [ 658]
- Selecting 3/4-axis round/profile tube machining [FCT-M5]

Lateral surface machining
- deselecting
p. [ 658]
- Deselecting 3/4-axis round/profile tube machining [FCT-M5]

5/6-axis round tube machining
5/6-axis profile tube machining
[FCT-M5]
[FCT-M5]

D
```

\#DELETE
\#DEL DIST2GO
\#DISABLE MODAL CYCLE
\#DIST TO GO BEGIN/END
\#DISTANCE PROG START ON/
OFF/ CLEAR
\#DRIVE WR SYN
\#DRIVE WAIT SYN

```
\＃DYNAMIC WEIGHT ON／OFF

E
\＃ECS ON／OFF
\＃EDGE MACHINING ON／OFF
\＃ENABLE AX LINK
\＃DISABLE AX LINK
\＃ERROR
\＃EXPL SYN
\＃EXPORT VE
\＃EXTCOMP ON／OFF

F
\＃FACE
\＃FACE OFF
\＃FF
\＃FGROUP
\＃FGROUP ROT
\＃FGROUP WAXIS
\＃FILE NAME
\＃FILE RENAME
\＃FILE DELETE
\＃FILE EXIST
\＃FILTER ON／OFF
\＃FLUSH
\＃FLUSH CONTINUE
\＃FLUSH WAIT
\＃FRC
\＃FREE TOOL CHANGE ON／OFF

Deleting self－defined variables or parameters
p．［ 602］
Deleting distance to go
Disabling a modal cycle
Distance to go display in a program section
Covered distance from program start
p．［ 490\(]\)
［FCT－C6］
Switching drive functions：Synchronous writing
p［ 461］
Switching drive functions：Synchronous waiting for acknow－
p［ 461］ ledgement
Selecting／deselecting dynamic weighting
p．［＞482］

Selecting／deselecting effector coordinate system p．［＞741］
Selecting／deselecting edge machining p．［ 480］
Selecting／deselecting axis couplings p．［户 362］

User－defined error output p．［ 417］
Explicit synchronisation p．［户 690］
Exporting V．E．variables in structures for a PLC integration［FCT－C22］
Selecting／deselecting external compensation
［FCT－C38］

Selecting face machining p．［〉 652］
Deselecting face machining p．［ 654］
Weighting of external feedrate p．［＞483］
Defining a feedrate group p．［ 4 422］
Calculating feedrate for rotary axes p．［＞422］
Weakest axis as feed axis p．［户 422］
Definition of file names p．［户 434］
Renaming a file p．［ \({ }^{2} 436\) ］
Deleting a file p．［户 438］
Checking the existence of a file p．［ 439］
Selecting／deselecting FIR filters and parameterisation p．［ 257］
Flushing NC channel with interrupted motion p．［户 339］
Flushing NC channel with continuous motion p．［ 339］
Synchronising decoding and interpolation p．［ 339］
Inserting chamfers and roundings：feedrate in chamfer or p．［ 158］
rounding segment
Tool change with active synchronous mode
p．［ 457\(]\)

\section*{G}
\＃GANTRY OFF
\＃GANTRY OFF ALL
\＃GANTRY ON
\＃GANTRY ON ALL
\＃GEAR LINK ON／OFF
\＃GET CMDPOS
\＃GET ACTPOS
\＃GET MANUAL OFFSETS
\＃GET WCS POSLIMIT

Disabling gantry combination p．［〉 485］
Disabling all gantry combinations p．［ 485］
Restoring a gantry combination p．［＞485］
Restoring all gantry combinations p．［＞485］
Selecting／deselecting position controller－based axis couplings p．［＞486］ and parameterisation
Requesting and storing current command positions of axes p．［〉 182］
Requesting and storing current actual positions of axes p．［＞183］
Requesting and storing current manual mode offsets
p．［ 178］
Auxiliary function to calculate motion limits in the workpiece co－
p．［ 744］

Setting handwheel parameters p．［＞172］
Contouring with short blocks p．［ 250］
－Smoothing methods p．［＞247］
－Trimming a contour p．［ 250］
－SURFACE optimiser p．［＞253］

Non－synchronised write of SERCOS parameters p．［＞374］
Synchronised write of SERCOS parameters p．［ 376］
Non－synchronised read of SERCOS parameters p．［ 375］
Initialising macro table
p．［ 709］
Initialising V．E．variables p．［〉 615］

Parameterising continuous jog mode
p．［ 173］
Parameterising incremental jog mode
p．［ 174］

Selecting machine kinematics
p．［ 720］
Modify kinematic characteristics
p．［ 721］

Settings for look－ahead
［FCT－C45］
Loading or restoring a saved configuration
p．［ 308］
Refresh all kinematic parameters of the TCP kinematic

M
\#MACHINE DATA
\#MAIN SPINDLE
\#MANUAL LIMITS
\#MCS ON/OFF
\#MCS TO WCS
\#MEAS MODE
\#MEAS
\#MEAS DEFAULT
\#MSG
\#MSG INFO
\#MSG SAVE

N
\#NIBBLE ON/OFF

0
\#OPTIONAL EXECUTION ON/ OFF
\#ORI MODE

\section*{\#OTC ON/OFF \\ \#OVERRIDE}

Writing machine data
Changing the main spindle
Specifying offset limits in manual mode
Temporary transition to the machine coordinate system
Mapping machine coordinates into workpiece coordinates
Switching measurement type
Extended programming of measurement options
Resetting extended measurement options
Sending a message from the NC program
Sending a message with additional information for the receiver
Writing a message to a file

Selecting/deselecting nibbling

Skipping program sequences during forward/backward motion
p [ 453]

Orientation programming
- Programming and configuration of 5-axis kinematics p. [> 751]
- Programming and configuration of 6 -axis kinematics

Online tool compensation
Programmable path override

Selecting position preset
p. [ 353]

Deselecting position preset p. [ 353]
Selecting/deselecting positioning without compensation motion
p. [ 722]

Selecting/deselecting punching
p [ 473]
Releasing axes
p [ 317]
Releasing all axes p [ 317]

Inserting chamfers and roundings: defining rounding
p. [ 158]

Select/deselect contour rotation
p. [ 398]

Definition of real-time cycle in NC program
kernel Industrielle Steuerungstechnik GmbH
```

S
\#SAVE CONFIG
\#SCALE ON / OFF
\#SEGMENTATION ON/OFF/ALL
\#SET AX
\#SET AX LINK
\#SIGNAL
\#SIGNAL REMOVE
\#SIGNAL READ
\#SINGLE STEP
\#SLOPE
\#SLOPE DEFAULT
\#SPLINE ON
\#SPLINE OFF
\#SPLINE TYPE AKIMA
\#SPLINE TYPE BSPLINE
\#STOP REVERSIBLE
\#STROKE DEF BEGIN/END
\#SUPPRESS OFFSETS

```
```

T

```
T
#TANGFEED
#TANGFEED
#TIME
#TIME
#TIMER
#TIMER
#TLAX
#TLAX
#TLAX DEFAULT
#TLAX DEFAULT
#TLC ON/OFF
#TLC ON/OFF
#TOOL DATA
#TOOL DATA
#TOOL LIFE READ/REMOVE
#TOOL LIFE READ/REMOVE
#TOOL ORI CS
#TOOL ORI CS
#TOOL PREP
#TOOL PREP
#TOOL REFRESH
#TOOL REFRESH
#TRACK CS ABS/ON/OFF
#TRACK CS ABS/ON/OFF
#TRAFO ON / OFF
#TRAFO ON / OFF
#TRANSVELMIN ON/OFF
#TRANSVELMIN ON/OFF
#TRC
#TRC
#TURN
```

\#TURN

```
\begin{tabular}{|c|c|}
\hline Saving current configuration & p. [ 305] \\
\hline Enlarging and reducing contours & p. [ 465] \\
\hline Selecting/deselecting segmentation of linear and circular blocks & p [> 463] \\
\hline Define an axis configuration & p [ 319] \\
\hline Programming axis couplings & p. [ 355] \\
\hline Sending signals & p [ 385] \\
\hline Clearing broadcast signals & p. [ 387] \\
\hline Reading signals without waiting & p. [ 391] \\
\hline Locking program areas for single-step mode & p [ 458] \\
\hline Parameterising the acceleration profile & p [ 372] \\
\hline Initial state of acceleration profile & p. [ 372] \\
\hline \multicolumn{2}{|l|}{Selecting spline interpolation} \\
\hline - Selecting Akima spline interpolation & p. [ 291] \\
\hline - Selecting B spline interpolation & p. [ 297] \\
\hline \multicolumn{2}{|l|}{Deselecting spline interpolation} \\
\hline - Deselecting Akima spline interpolation & p [ \({ }^{\text {2 }}\) 292] \\
\hline - Deselecting B spline interpolation & p [ 298] \\
\hline Selecting Akima spline & p [ 291] \\
\hline Selecting B spline & p [ 297] \\
\hline Defining stop flags for forward/backward motion & [FCT-C7] \\
\hline Defining stroke motion for punching/nibbling & p [ \({ }^{\text {P 473] }}\) \\
\hline Suppressing offsets & p [ 347\(]\) \\
\hline
\end{tabular}

Adapting minimum radius for tangential feed Dwell time p. [ 345]Time measurementp. [ 83]
Free assignment of tool length correction in an axis ..... p. [ 495]p [ 420]
Assigning tool length correction to the 3rd main axis Assigning tool length correction to the 3rd main axis ..... p. [ 495]
Selecting/deselecting tool length compensation ..... p. [ 715]
Requesting tool data ..... p [ 770]
Reading/removing tool life data ..... p. [ 772]
Orienting tool ..... p [ 718]
Preparing for a tool change ..... p [ 770]
Refreshing tool data ..... p [ 773]
Tracking dynamic CS ..... [FCT-C30]
Selecting/deselecting a kinematic transformation ..... p. [ 713]
Selecting/deselecting minimum block transition velocity ..... p. [ 429]
Programmable additional options of TRC ..... p. [ 564]
Settings for turning functionsp. [ 489]

\section*{V \\ \#VAR...\#ENDVAR \\ \#VECTOR LIMIT ON/OFF \\ \#VIB GUARD \\ \#VOLCOMP ON/OFF}

Declaration block for self-defined variables or parameters
- Variables
p. [ \(\left.{ }^{[ } 602\right]\)
- Parameter
p. [ 224]

Adapting path dynamic limit values
p. [ 425]

Activating Vibration Guard
[FCT-C36]
Selecting/deselecting volumetric compensation
[FCT-C26]
w
\#WAIT
\#WAIT FOR
\#WAIT INDP
\#WAIT INDP ALL
\#WCS TO MCS
Waiting for signals
p [ 389]
Waiting for an event
p [ 344]
Waiting for asynchronous independent axis
p [ 774]
Waiting for all asynchronous independent axes
p [ 773]
Mapping workpiece coordinates in machine coordinates
p [ 744]
\begin{tabular}{|c|c|c|}
\hline \multicolumn{3}{|l|}{22.1.6 Additional axis-specific functions (<X>[..])} \\
\hline INDP_SYN & Synchronous (blockwise) independent axis motion & p [ \({ }^{\text {7 7 }}\) 74] \\
\hline INDP_ASYN & Asynchronous (cross-block) independent axis motion & p [ 774 ] \\
\hline OSC & Oscillating axes & p. [ \({ }^{\text {P }} 778\) ] \\
\hline COMP & Selecting/deselecting axis compensation & p. [ \({ }^{\text {P 7 }}\) 74] \\
\hline DIST_CTRL & Distance control (sensed spindles) & p. [ \({ }^{\text {P 7 7 }}\) ] \\
\hline OVERRIDE & Programmable axis override & p [ 790] \\
\hline DYNAMIC & Programmable acceleration overload & p [ 791] \\
\hline \begin{tabular}{l}
LIFT \\
LIFT_START / LIFT_END
\end{tabular} & Lifting/lowering an axis & p [ \({ }^{\text {P 797] }}\) \\
\hline SYNC IN/OUT & Synchronising an axis in coordinated motion & p [ 792] \\
\hline POLY & Programming an axis polynomial & p [ 794] \\
\hline
\end{tabular}

\subsection*{22.1.7 PLC-Open functions (<X>[MC_..])}
\begin{tabular}{|c|c|c|}
\hline MC_Home & Homing & S. [ \({ }^{\text {c }}\) 697] \\
\hline MC_MoveAbsolute & Axis motion to an absolute position & S. [ 898\(]\) \\
\hline MC_MoveAdditive & Relative axis motion to the commanded position & S. [ 699] \\
\hline MC_MoveRelative & Relative axis motion to the current position & S. [ 700\(]\) \\
\hline MC_MoveSuperlmposed & Relative axis motion to a motion already active & S. [ 701 ] \\
\hline MC_MoveVelocity & Endless axis motion at the specified velocity & S. [ 702\(]\) \\
\hline MC_Stop & Stop an axis motion & S. [ 703\(]\) \\
\hline MC_Gearln & Gear coupling with a gear ratio & S. [ \({ }^{\text {P }} 704\) ] \\
\hline MC_GearOut & Release a gear coupling & S. [ 706\(]\) \\
\hline MC_Phasing & Phase offset of couplings & S. [ \({ }^{\text {P }} 707\) ] \\
\hline MC_TouchProbe & Measure an axis position & B [ 708] \\
\hline
\end{tabular}

\subsection*{22.1.8 Variable programming (V.)}
V.A. ... Axis-specific variables S. [ 578]
V.SPDL. ... Spindle-specific variables S. [ 582]
V.SPDL_PROG. ...
V.G. ... Global variables S. [ 584]
V.P. ... Self-defined variables, program global S. [ 605]
V.S. ... Self-defined variables, (main) program global S. [ 607]
V.L. ... Self-defined variables, program local
S. [ 609]
V.E. ...

External variables
S. [ 611]
V.CYC. ... User-defined variables, cycle variables
S. [ 615]
V.TOOL. ...

Tool identification variables
S. [ 770\(]\)
V.TLM. ...

Tool life variables
S. [ 771]

\subsection*{22.1.9 Miscellaneous functions}

Various calculation operations and functions are provided to program mathematic expressions [> 31] (e.g. SIN, COS, MOD, ABS, OR..) and to process strings [> 36] (e.g. LEFT, MID, INSERT..).

\subsection*{22.1.10 Migrated NC commands}

\section*{Release Note}

The following table lists commands which were transferred to a new NC syntax as a result of functional advanced developments or for syntax reasons.
Commands previously used may still be used for programming (downwards compatibility) but they should no longer be used to generate new NC programs.
OId syntax:
\#SET DEC LR SOLL
\#VECTORVEL ON / OFF
\#VECTORACC ON / OFF
\#INIT MAKRO TAB
G200 \#ACHSE [...]
G201 \#ACHSE [..]
G202 \#ACHSE [..]]
\#GET IPO OFFSET
\#SET OFFSET [...] X
\#SET HR [...] X
\#SET TIP [...] X
\#SET JOG [...] X
\#SET IPO SOLLPOS [...]
\#SET SLOPE PROFIL [...]
\#SET ASPLINE STARTTANG X.. Y..
\#SET ASPLINE ZIELTANG X.. Y..
\#SET ASPLINE MODE [...]
\#SET CORNER PARAM [...]
\#SET TANGFEED RMIN [...]
\#SET SPLINE ON / OFF
\#SET SPLINETYPE AKIMA
\#SET SPLINETYPE BSPLINE
\#RTCP ON / OFF
\begin{tabular}{|c|c|}
\hline New syntax & from version \\
\hline \#CHANNEL INIT [...] [ 179] & V2.10.1504.00 \\
\hline \#VECTOR LIMIT ON / OFF [...] [ [ 425] & V2.10.1507.02 \\
\hline \#VECTOR LIMIT ON / OFF [...] [ \({ }^{\text {4 425] }}\) & V2.10.1507.02 \\
\hline \#INIT MACRO TAB [ \({ }^{\text {7 709] }}\) & V2.11.2010.02 \\
\hline G200 X.. Y.. [ [ 171] & V2.11.2010.02 \\
\hline G201 X.. Y.. [ 169\(]\) & V2.11.2010.02 \\
\hline G202 X.. Y.. [> 169] & V2.11.2010.02 \\
\hline \#GET MANUAL OFFSETS [ \({ }^{\text {P 178] }}\) & V2.11.2010.02 \\
\hline \#MANUAL LIMITS [...][> 174] & V2.11.2010.02 \\
\hline \#HANDWHEEL [...][> 172] & V2.11.2010.02 \\
\hline \#JOG CONT [...][ 173] & V2.11.2010.02 \\
\hline \#JOG INCR [...][ 174] & V2.11.2010.02 \\
\hline \#GET CMDPOS [...][ 182] & V2.11.2010.02 \\
\hline \#SLOPE ... [...] [ 372] & V2.11.2010.02 \\
\hline \#AKIMA STARTVECTOR X.. Y.. [〉 294] & V2.11.2010.02 \\
\hline \#AKIMA ENDVECTOR X.. Y.. [〉 294] & V2.11.2010.02 \\
\hline \#AKIMA TRANS [...] [ 293\(]\) & V2.11.2010.02 \\
\hline \#CORNER PARAM [...] [ 130] & V2.11.2010.02 \\
\hline \#TANGFEED [...] [ \({ }^{\text {345] }}\) & V2.11.2010.02 \\
\hline \#SPLINE ON [ 291]/ OFF [ 292\(]\) & V2.11.2010.02 \\
\hline \#SPLINE TYPE AKIMA [ 291\(]\) & V2.11.2010.02 \\
\hline \#SPLINE TYPE BSPLINE [ \({ }^{\text {297] }}\) & V2.11.2010.02 \\
\hline \#TRAFO ON / OFF [ \({ }^{\text {P13] }}\) & all versions \\
\hline
\end{tabular}

\subsection*{22.2 Revision history}
\begin{tabular}{|c|c|c|}
\hline Version & Entry & Date \\
\hline 1.0 & First version of programming manual in new layout & 01/10/2019 \\
\hline 1.01 & TRC: Direct selection/deselection (G236) of TRC directly on the path & 07.11.2019 \\
\hline 1.02 & Set an axis position in the channel & 20.03.2020 \\
\hline 1.03 & TRC option: PERPENDICULAR_RADIUS_CHANGE and STRETCH_FACTOR & 24.03.2020 \\
\hline 1.13 & Section V.CYC: new & 01.04.2020 \\
\hline 1.14 & Several minor content and editorial errors rectified & 02.04.2020 \\
\hline 1.15 & Integrate \#FF & 22.05.2020 \\
\hline & Restrictions with NC filenames & 27.05.2020 \\
\hline 1.16 & \#CONTOUR MODE -> PATH_DIST new & 08.07.2020 \\
\hline & V.G. variables : INVOKE_COUNT and LIST_COUNT new & 08.07.2020 \\
\hline 1.17 & Axis-specific programming "Move to fixed stop" new & 26.08.2020 \\
\hline 1.18 & Exception list of commands with active TRC/SRK in TRC section & 26.11.2020 \\
\hline 1.20 & \#GANTRY ON / OFF integrated & 01.02.2021 \\
\hline 1.211 & \#ORI MODE in overview of commands & 03.02.2021 \\
\hline 1.212 & V.G.CAXTRACK_ACTIVE new. & 05.02.2021 \\
\hline 1.213 & LENGTH_LONG_CIR in \#HSC [ SURFACE..] new & 15.03.2021 \\
\hline 1214 & Exchange filter programming by "Program FIR filters" in smoothing methods & 12.04.2021 \\
\hline 1.215 & Adapt/supplement several headings to the corresponding programming command according to "Overview of programming commands". & 23.04.2021 \\
\hline 1.216 & Integration of \#KIN DATA[] & 16.09.2021 \\
\hline 1.217 & Several minor content and editorial errors rectified, Macroprogramming section supplemented by dynamic configuration, Appendix section Additional functions extended by quick-access letter bar. Letter ranges fragmented. & \[
\begin{aligned}
& \text { 23.05.2022, } \\
& \mathrm{Gr}
\end{aligned}
\] \\
\hline 1.218 & \#DISABLE MODAL CYCLE supplemented with link in the appendix. & 29.06.2022 \\
\hline 1.219 & \#DIST TO GO BEGIN/END integrated & 02.08.2022 \\
\hline
\end{tabular}

\section*{23 References}
[1] Documentation/General description of channel parameters [CHAN]
\begin{tabular}{|l|l|}
\hline No. & Description \\
\hline 1 & Elements of the structure makro_def[i].* \\
\hline 2 & Elements of the structure synchro_data.koppel_gruppe[0].* \\
\hline 3 & Elements of the structure spindel[i].* \\
\hline 4 & Elements of the structure spindel[i].range_table[i].* \\
\hline 5 & Elements of the structure gruppe[j].achse[i].* \\
\hline 6 & Elements of the structure speed_limit_look_ahead.* \\
\hline 7 & Elements of the structure dynamic_weighting[i].* \\
\hline
\end{tabular}
[2] Documentation/General description of axis parameters [AXIS]
\begin{tabular}{|l|l|}
\hline No. & Description \\
\hline 1 & Elements of the structure getriebe[i].slope_profil.* \\
\hline 2 & Elements of the structure getriebe[i].Islope_profil.* \\
\hline 3 & Elements of the structure filter[i].* \\
\hline
\end{tabular}
[3] Documentation/General description of zero offset data [ZERO]
[4] Specifications of SERCOS Interface, IEC 61491
[5] Documentation/General description of tool data [TOOL]
[6] Documentation of control/manufacturer specific settings of system parameters [SYSP]
[7] Documentation/General description of start-up list [STUP]
[8] Documentation/General description of external variables [EXTV]
[9] Motion Control Platform for PLCopen [MCP-P1]
© Copyright
ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart
All rights reserved
www.isg-stuttgart.de```


[^0]:    S[MC_MoveAdditive Distance=277 Velocity=1100 Acceleration=550 \}
    Deceleration=660 Jerk=22000]

[^1]:    S[MC MoveSuperImposed Distance=321 VelocityDiff=783 Acceleration=811 \}
    Deceleration=922 Jerk=45000]

[^2]:    S[MC_Phasing Master=11 PhaseShift=25 Velocity=1000 Acceleration=500 \
    Deceleration=600 Jerk=20000]

