### SINUMERIK® Documentation

#### Printing history

Brief details of this edition and previous editions are listed below.

The status of each edition is shown by the code in the "Remarks" column.

**Status code in the "Remarks" column:**

- **A ....** New documentation.
- **B ....** Unrevised reprint with new Order No.
- **C ....** Revised edition with new status.

If factual changes have been made on the page since the last edition, this is indicated by a new edition coding in the header of that page.

<table>
<thead>
<tr>
<th>Edition</th>
<th>Order No.</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>02.95</td>
<td>6FC5298-2AB00-0BP0</td>
<td>A</td>
</tr>
<tr>
<td>04.95</td>
<td>6FC5298-2AB00-0BP1</td>
<td>C</td>
</tr>
<tr>
<td>12.95</td>
<td>6FC5298-3AB10-0BP0</td>
<td>C</td>
</tr>
<tr>
<td>03.96</td>
<td>6FC5298-3AB10-0BP1</td>
<td>C</td>
</tr>
<tr>
<td>08.97</td>
<td>6FC5298-4AB10-0BP0</td>
<td>C</td>
</tr>
<tr>
<td>12.97</td>
<td>6FC5298-4AB10-0BP1</td>
<td>C</td>
</tr>
<tr>
<td>12.98</td>
<td>6FC5298-5AB10-0BP0</td>
<td>C</td>
</tr>
<tr>
<td>08.99</td>
<td>6FC5298-5AB10-0BP1</td>
<td>C</td>
</tr>
<tr>
<td>04.00</td>
<td>6FC5298-5AB10-0BP2</td>
<td>C</td>
</tr>
<tr>
<td>10.00</td>
<td>6FC5298-6AB10-0BP0</td>
<td>C</td>
</tr>
<tr>
<td>09.01</td>
<td>6FC5298-6AB10-0BP1</td>
<td>C</td>
</tr>
<tr>
<td>11.02</td>
<td>6FC5298-6AB10-0BP2</td>
<td>C</td>
</tr>
</tbody>
</table>

This manual is included in the documentation on CD-ROM (DOCONCD)

<table>
<thead>
<tr>
<th>Edition</th>
<th>Order No.</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.02</td>
<td>6FC5298-6CA00-0BG2</td>
<td>C</td>
</tr>
</tbody>
</table>

#### Trademarks

SIMATIC®, SIMATIC HMI®, SIMATIC NET®, SIROTEC®, SINUMERIK® and SIMODRIVE® are registered trademarks of Siemens AG. The other designations in this publication may also be trademarks, the use of which by third parties may constitute copyright violation.

Further information is available on the Internet under: http://www.ad.siemens.de/sinumerik

This publication was produced with WinWord V8.0 and Designer V7.0. The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

© Siemens AG, 1995–2001. All rights reserved

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

We have checked that the contents of this documentation correspond to the hardware and software described. Nonetheless, differences might exist and we cannot therefore guarantee that they are completely identical. The information contained in this document is, however, reviewed regularly and any necessary changes will be included in the next edition. We welcome suggestions for improvement.

Subject to change without prior notice.
Contents

Preface 0-14

Flexible NC Programming 1-25

1.1 Variable and arithmetic parameters ................................................................. 1-26
1.2 Variable definition ............................................................................................. 1-29
1.3 Array definition .................................................................................................. 1-34
1.4 Indirect programming ........................................................................................ 1-40
1.5 Assignments ....................................................................................................... 1-45
1.6 Arithmetic operations and functions ................................................................... 1-46
1.7 Comparison and logic operators ......................................................................... 1-48
1.8 Priority of operators ........................................................................................... 1-53
1.9 Possible type conversions .................................................................................. 1-54
1.10 String operations .............................................................................................. 1-55
1.10.1 Type conversion ............................................................................................ 1-56
1.10.2 Concatenation of strings ............................................................................... 1-58
1.10.3 Conversion to lower/upper case .................................................................... 1-59
1.10.4 Length of the string ..................................................................................... 1-60
1.10.5 Search for character/string in a string ......................................................... 1-60
1.10.6 Selection of a substring ............................................................................... 1-62
1.10.7 Selection of a single character ..................................................................... 1-63
1.11 CASE instruction ............................................................................................. 1-65
1.12 Control structures ............................................................................................ 1-67
1.13 Program coordination ....................................................................................... 1-72
1.14 Interrupt routine .............................................................................................. 1-77
1.15 Axis transfer, spindle transfer .......................................................................... 1-85
1.16 NEWCONF: Setting machine data active (SW 4.3 and higher) ....................... 1-90
1.17 WRITE: Write file (SW 4.3 and higher) ......................................................... 1-91
1.18 DELETE: Delete file (SW 4.3 and higher) ....................................................... 1-93
1.19 READ: Read lines in file (SW 5.2 and higher) ............................................... 1-94
1.20 ISFILE: File available in user memory NCK (SW 5.2 and higher) ................... 1-97
1.21 CHECKSUM: Creation of a checksum over an array (SW 5.2 and higher) ...... 1-98
## Subprograms, Macros

2-101

2.1 Using subprograms ................................................................. 2-102

2.2 Subprogram with SAVE mechanism ........................................ 2-104

2.3 Subprograms with parameter transfer ...................................... 2-105

2.4 Calling subprograms: L or EXTERN ....................................... 2-109

2.5 Parameterizable subprogram return (SW 6.4 and higher) .......... 2-113

2.6 Subprogram with program repetition: P .................................. 2-117

2.7 Modal subprogram: MCALL ................................................... 2-118

2.8 Calling the subprogram indirectly: CALL ............................... 2-119

2.9 Repeating program sections with indirect programming (SW 6.4 and higher) ................. 2-120

2.10 Calling up a program in ISO language indirectly: ISOCALL ........ 2-121

2.11 Calling subprogram with path specification and parameters PCALL ........................................ 2-122

2.12 Extending a search path for subprogram calls with CALLPATH (SW 6.4 and higher) 2-123

2.13 Suppress current block display: DISPLOF ............................. 2-125

2.14 Single block suppression: SBLOF, SBLON (SW 4.3 and higher) .......... 2-126

2.15 Executing external subprogram: EXTCALL (SW 4.2 and higher) 2-132

2.16 Subprogram call with M/T function ......................................... 2-135

2.17 Cycles: Setting parameters for user cycles ............................. 2-138

2.18 Macros. DEFINE...AS .......................................................... 2-142

## File and Program Management

3-145

3.1 Overview ................................................................................. 3-146

3.2 Program memory ................................................................. 3-147

3.3 User memory .......................................................................... 3-153

3.4 Defining user data ................................................................. 3-156

3.5 Defining protection levels for user data (GUD) ......................... 3-160

3.6 Automatic activation of GUDs and MACs (SW 4.4 and higher) ................. 3-162

3.7 Data-specific protection level change for machine and setting data ........................................ 3-164

3.7.1 Change .............................................................................. 3-164

3.7.2 Undoing a change ............................................................. 3-165

3.8 Changing attributes of NC language elements (SW 6.4 and higher) ......................... 3-165

3.9 Structuring instruction SEFORM in the Step editor (SW 6.4 and higher) ......................... 3-173
Contents

Protection Zones 4-175
  4.1 Defining the protection zones CPROTDEF, NPROTDEF ........................................ 4-176
  4.2 Activating/deactivating protection zones: CPROT, NPROT ...................................... 4-180

Special Motion Commands 5-185
  5.1 Approaching coded positions, CAC, CIC, CDC, CACP, CACN .................................... 5-186
  5.2 Spline interpolation ........................................................................................................ 5-187
  5.3 Compressor COMPON/COMPCURV/COMPCAD (SW 6.2) ........................................... 5-196
  5.4 Polynomial interpolation – POLY, POLYPATH (SW 5 and higher) ................................ 5-204
  5.5 Settable path reference, SPATH, UPATH (SW 4.3 and higher) ...................................... 5-211
  5.6 Measurements with touch trigger probe, MEAS, MEAW ............................................. 5-215
  5.7 Extended measuring function MEASA, MEAWA, MEAC (SW 4 and higher, option) ....... 5-218
  5.8 Special functions for OEM users .................................................................................. 5-228
  5.9 Programmable motion end criterion (SW 5.1 and higher) ........................................... 5-229
  5.10 Programmable servo parameter block (SW 5.1 and higher) ....................................... 5-232

Frames 6-235
  6.1 Coordinate transformation via frame variables ............................................................ 6-236
  6.2 Frame variables/assigning values to frames ................................................................. 6-241
  6.3 Coarse/fine offset ......................................................................................................... 6-248
  6.4 DRF offset .................................................................................................................... 6-249
  6.5 External zero offset ..................................................................................................... 6-250
  6.6 Programming PRESET offset, PRESETON ................................................................. 6-251
  6.7 Deactivating frames ..................................................................................................... 6-252
  6.8 Frame calculation from three measuring points in the area: MEAFRAME ...................... 6-253
  6.9 NCU-global frames (SW 5 and higher) ........................................................................ 6-256
    6.9.1 Channel-specific frames ........................................................................................ 6-257
    6.9.2 Frames active in the channel ................................................................................ 6-259

Transformations 7-265
  7.1 Three, four and five axis transformation: TRAORI ....................................................... 7-266
    7.1.1 Programming tool orientation ................................................................................ 7-269
    7.1.2 Orientation axes reference – ORIWCS, ORIMCS .................................................. 7-274
    7.1.3 Singular positions and how to handle them .......................................................... 7-275
    7.1.4 Orientation axes (SW 5.2 and higher) .................................................................. 7-276
Contents

7.1.5 Cartesian PTP travel (from SW 5.2) .............................................................. 7-279
7.1.6 Online tool length compensation (SW 6.4 and higher) ........................... 7-284
7.2 Milling turned parts: TRANSMIT ................................................................. 7-287
7.3 Cylinder surface transformation: TRACYL .................................................... 7-290
7.4 Inclined axis: TRAANG .................................................................................. 7-296
7.4.1 Inclined axis programming: G05, G07 (SW 5.3 and higher) ..................... 7-300
7.5 Constraints when selecting a transformation ................................................. 7-302
7.6 Deselect transformation: TRAFOOF ............................................................ 7-304
7.7 Chained transformations ............................................................................... 7-305
7.8 Switchable geometry axes, GEOAX ............................................................. 7-308

Tool Offsets 8-313

8.1 Offset memory ............................................................................................... 8-314
8.2 Language commands for tool management .................................................. 8-316
8.3 Online tool offset PUTFTOCF, PUTFTOC, FTOCON, FTOCOF ................... 8-319
8.4 Maintain tool radius compensation at constant level, CUTCONON (SW 4 and higher) ................................................................. 8-325
8.5 Activate 3D tool offsets ................................................................................. 8-328
8.6 Tool orientation .............................................................................................. 8-336
8.7 Free assignment of D numbers, cutting edge number CE (SW 5 and higher) 8-341
8.7.1 Check D numbers (CHKDNO) ................................................................. 8-342
8.7.2 Renaming D numbers (GETDNO, SETDNO) ............................................. 8-343
8.7.3 T numbers for the specified D number (GETACTTD) ............................... 8-344
8.7.4 Set final D numbers to invalid ................................................................... 8-345
8.8 Toolholder kinematics .................................................................................. 8-346

Path Traversing Behavior 9-351

9.1 Tangential control TANG, TANGON, TANGOF, TANGDEL ............................ 9-352
9.2 Coupled motion TRAILON, TRAILOF ......................................................... 9-358
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV, CTABSSV, CTABSEV ................................................................. 9-362
9.4 Axial leading value coupling, LEADON, LEADOF ........................................ 9-375
9.5 Feed characteristic, FNORM, FLIN, FCUB, FPO ........................................ 9-381
9.6 Program run with preprocessing memory, STARTFIFO, STOPFIFO, STOPRE 9-386
9.7 Repositioning on contour, REPOSA, REPOS L, REPOSQ, REPOS H ........... 9-388
## Motion-Synchronous Action

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 Structure, basic information</td>
<td>10-395</td>
</tr>
<tr>
<td>10.1.1 Programming and command elements</td>
<td>10-397</td>
</tr>
<tr>
<td>10.1.2 Validity range: Identification number ID</td>
<td>10-398</td>
</tr>
<tr>
<td>10.1.3 Vocabulary word</td>
<td>10-399</td>
</tr>
<tr>
<td>10.1.4 Actions</td>
<td>10-402</td>
</tr>
<tr>
<td>10.1.5 Overview of synchronized actions</td>
<td>10-404</td>
</tr>
<tr>
<td>10.2 Basic modules for conditions and actions</td>
<td>10-406</td>
</tr>
<tr>
<td>10.3 Special real-time variables for synchronized actions</td>
<td>10-409</td>
</tr>
<tr>
<td>10.3.1 Flags/counters $AC_MARKER[n]$</td>
<td>10-409</td>
</tr>
<tr>
<td>10.3.2 Timer variable $AC_TIMER[n]$, SW 4 and higher</td>
<td>10-409</td>
</tr>
<tr>
<td>10.3.3 Synchronized action parameters $AC_PARAM[n]$</td>
<td>10-410</td>
</tr>
<tr>
<td>10.3.4 Access to R parameters $Rxx$</td>
<td>10-411</td>
</tr>
<tr>
<td>10.3.5 Machine and setting data read/write (SW 4 and higher)</td>
<td>10-412</td>
</tr>
<tr>
<td>10.3.6 FIFO variable $AC_FIFO1[n] … $AC_FIFO10[n]$ (SW 4 and higher)</td>
<td>10-413</td>
</tr>
<tr>
<td>10.4 Actions within synchronized actions</td>
<td>10-415</td>
</tr>
<tr>
<td>10.4.1 Auxiliary functions output</td>
<td>10-415</td>
</tr>
<tr>
<td>10.4.2 Set read-in disable RDISABLE</td>
<td>10-416</td>
</tr>
<tr>
<td>10.4.3 Cancel preprocessing stop STOPREOF</td>
<td>10-417</td>
</tr>
<tr>
<td>10.4.4 Deletion of distance-to-go</td>
<td>10-418</td>
</tr>
<tr>
<td>10.4.5 Delete distance-to-go with preparation, DELDTG, DELEDTG (&quot;Axis 1 to x&quot;)</td>
<td>10-418</td>
</tr>
<tr>
<td>10.4.6 Polynomial definition, FCTDEF, block-synchronized</td>
<td>10-420</td>
</tr>
<tr>
<td>10.4.7 Laser power control</td>
<td>10-422</td>
</tr>
<tr>
<td>10.4.8 Evaluation function SYNFCT</td>
<td>10-423</td>
</tr>
<tr>
<td>10.4.9 Adaptive control (additive)</td>
<td>10-424</td>
</tr>
<tr>
<td>10.4.10 Adaptive control (multiplicative)</td>
<td>10-425</td>
</tr>
<tr>
<td>10.4.11 Clearance control with limited compensation</td>
<td>10-426</td>
</tr>
<tr>
<td>10.4.12 Online tool offset FTOC</td>
<td>10-428</td>
</tr>
<tr>
<td>10.4.13 Positioning movements</td>
<td>10-430</td>
</tr>
<tr>
<td>10.4.14 Position axis POS</td>
<td>10-432</td>
</tr>
<tr>
<td>10.4.15 Start/stop axis MOV</td>
<td>10-432</td>
</tr>
<tr>
<td>10.4.16 Axial feed FA</td>
<td>10-433</td>
</tr>
<tr>
<td>10.4.17 SW limit switch</td>
<td>10-434</td>
</tr>
<tr>
<td>10.4.18 Axis coordination</td>
<td>10-434</td>
</tr>
<tr>
<td>10.4.19 Set actual value</td>
<td>10-436</td>
</tr>
<tr>
<td>10.4.20 Spindle motions</td>
<td>10-437</td>
</tr>
<tr>
<td>10.4.21 Coupled-axis motion TRAILON, TRAILOF</td>
<td>10-438</td>
</tr>
<tr>
<td>10.4.22 Leading value coupling LEADON, LEADOF</td>
<td>10-439</td>
</tr>
<tr>
<td>10.4.23 Measurement</td>
<td>10-441</td>
</tr>
<tr>
<td>10.4.24 Set/clear wait marks: SETM, CLEARM (SW 5.2 and higher)</td>
<td>10-441</td>
</tr>
<tr>
<td>10.4.25 Error responses</td>
<td>10-442</td>
</tr>
<tr>
<td>10.4.26 Travel to fixed stop FXS and FOCON/FOCOF</td>
<td>10-442</td>
</tr>
</tbody>
</table>
10.5 Technology cycles ...................................................................................................... 10-445
10.5.1 Lock, unlock, reset: LOCK, UNLOCK, RESET .................................................. 10-447
10.6 Cancel synchronized action: CANCEL ..................................................................... 10-449
10.7 Supplementary conditions ...................................................................................... 10-450

Oscillation
11.1 Asynchronous oscillation ....................................................................................... 11-456
11.2 Oscillation controlled via synchronous actions ..................................................... 11-463

Punching and Nibbling
12.1 Activation, deactivation .......................................................................................... 12-476
12.1.1 Language commands........................................................................................... 12-476
12.1.2 Use of M commands ......................................................................................... 12-479
12.2 Automatic path segmentation ................................................................................. 12-480
12.2.1 Path segmentation for path axes ........................................................................ 12-481
12.2.2 Path segmentation for single axes ...................................................................... 12-482
12.2.3 Programming examples ..................................................................................... 12-484

Additional Functions
13.1 Axis functions AXNAME, SPI, ISAXIS, AXSTRING (SW 6 and higher)...................... 13-489
13.2 Function call ISVAR ( ) (SW 6.3 and higher)....................................................... 13-491
13.3 Learn compensation characteristics: QECLRNON, QECLRNOF .............................. 13-493
13.4 Synchronized spindle ............................................................................................. 13-495
13.5 EG: Electronic gear (SW 5 and higher) ................................................................... 13-505
13.5.1 Define electronic gear: EGDEF .......................................................................... 13-505
13.5.2 Activate electronic gear ..................................................................................... 13-506
13.5.3 Deactivate electronic gear ................................................................................ 13-510
13.5.4 Delete definition of an electronic gear ............................................................... 13-511
13.5.5 Revolutionsal feedrate (G95)/electronic gear (SW 5.2) ...................................... 13-511
13.5.6 Response of EG at Power ON, RESET, mode change, block search ............... 13-512
13.5.7 The electronic gear’s system variables ............................................................... 13-512
13.6 Extended stopping and retract (SW 5 and higher) .................................................. 13-513
13.6.1 Drive-independent reactions ............................................................................. 13-514
13.6.2 NC-controlled reactions ..................................................................................... 13-515
13.6.3 Possible trigger sources .................................................................................... 13-518
13.6.4 Logic gating functions: Source/reaction operation ........................................... 13-518
13.6.5 Activation .......................................................................................................... 13-519
13.6.6 Generator operation/DC link backup ............................................................... 13-519
13.6.7 Drive-independent stop ..................................................................................... 13-520
13.6.8 Drive-independent retract ................................................................. 13-521
13.6.9 Example: Using the drive-independent reaction .................................... 13-521
13.7 Link communication (SW 5.2 and higher) ................................................ 13-522
13.8 Axis container (SW 5.2 and higher) .......................................................... 13-526
13.9 Program execution time/Workpiece counter (SW 5.2 and higher) ............... 13-528
13.9.1 Program runtime .................................................................................. 13-528
13.9.2 Workpiece counter ............................................................................... 13-530
13.10 Interactive window call from parts program, command MMC
     (SW 4.4 and higher) .................................................................................. 13-532
13.11 Influencing the motion control ................................................................. 13-534
13.11.1 Percentage jerk correction: JERKLIM ............................................... 13-534
13.11.2 Percentage velocity correction: VELOLIM ........................................ 13-535
13.12 Master/slave grouping ............................................................................ 13-536
User Stock Removal Programs 14-541
14.1 Supporting functions for stock removal .................................................... 14-542
14.2 Contour preparation: CONTPRON ......................................................... 14-543
14.3 Contour decoding: CONTDCON (SW 5.2 and higher) .............................. 14-550
14.4 Intersection of two contour elements: INTERSEC ................................. 14-554
14.5 Traversing a contour element from the table: EXECTAB ...................... 14-556
14.6 Calculate circle data: CALCDAT ............................................................ 14-557
Tables 15-559
15.1 List of instructions .................................................................................... 15-561
15.2 List of system variables ............................................................................ 15-591
15.2.1 R parameters ....................................................................................... 15-591
15.2.2 Channel-specific synchronized action variables ................................... 15-591
15.2.3 Frames 1 ............................................................................................. 15-592
15.2.4 Toolholder data ................................................................................... 15-593
15.2.5 Channel-specific protection zones ....................................................... 15-601
15.2.6 Tool parameters .................................................................................. 15-603
15.2.7 Cutting edge data OEM user ............................................................... 15-609
15.2.8 Monitoring data for tool management ................................................ 15-617
15.2.9 Monitoring data for OEM users ........................................................... 15-618
15.2.10 Tool-related data ............................................................................... 15-619
15.2.11 Tool-related grinding data ................................................................. 15-621
15.2.12 Magazine location data ...................................................................... 15-622
15.2.13 Magazine location data for OEM users .............................................. 15-623
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.2.14</td>
<td>Magazine description data for tool management</td>
<td>15-624</td>
</tr>
<tr>
<td>15.2.15</td>
<td>Tool management magazine description data for OEM users</td>
<td>15-625</td>
</tr>
<tr>
<td>15.2.16</td>
<td>Magazine module parameter</td>
<td>15-626</td>
</tr>
<tr>
<td>15.2.17</td>
<td>Adapter data</td>
<td>15-626</td>
</tr>
<tr>
<td>15.2.18</td>
<td>Measuring system compensation values</td>
<td>15-626</td>
</tr>
<tr>
<td>15.2.19</td>
<td>Quadrant error compensation</td>
<td>15-627</td>
</tr>
<tr>
<td>15.2.20</td>
<td>Interpolatory compensation</td>
<td>15-629</td>
</tr>
<tr>
<td>15.2.21</td>
<td>NCK-specific protection zones</td>
<td>15-630</td>
</tr>
<tr>
<td>15.2.22</td>
<td>Cycle parameterization</td>
<td>15-631</td>
</tr>
<tr>
<td>15.2.23</td>
<td>System data</td>
<td>15-636</td>
</tr>
<tr>
<td>15.2.24</td>
<td>Frames 2</td>
<td>15-636</td>
</tr>
<tr>
<td>15.2.25</td>
<td>Tool data</td>
<td>15-638</td>
</tr>
<tr>
<td>15.2.26</td>
<td>Magazines</td>
<td>15-643</td>
</tr>
<tr>
<td>15.2.27</td>
<td>Programmed geometry axis values</td>
<td>15-646</td>
</tr>
<tr>
<td>15.2.28</td>
<td>G groups</td>
<td>15-647</td>
</tr>
<tr>
<td>15.2.29</td>
<td>Programmed values</td>
<td>15-647</td>
</tr>
<tr>
<td>15.2.30</td>
<td>Channel states</td>
<td>15-651</td>
</tr>
<tr>
<td>15.2.31</td>
<td>Synchronized actions</td>
<td>15-656</td>
</tr>
<tr>
<td>15.2.32</td>
<td>I/Os</td>
<td>15-657</td>
</tr>
<tr>
<td>15.2.33</td>
<td>Reading and writing PLC variables</td>
<td>15-657</td>
</tr>
<tr>
<td>15.2.34</td>
<td>NCU link</td>
<td>15-658</td>
</tr>
<tr>
<td>15.2.35</td>
<td>Direct PLC I/O</td>
<td>15-658</td>
</tr>
<tr>
<td>15.2.36</td>
<td>Tool management</td>
<td>15-659</td>
</tr>
<tr>
<td>15.2.37</td>
<td>Timers</td>
<td>15-662</td>
</tr>
<tr>
<td>15.2.38</td>
<td>Path movement</td>
<td>15-663</td>
</tr>
<tr>
<td>15.2.39</td>
<td>Speeds/accelerations</td>
<td>15-665</td>
</tr>
<tr>
<td>15.2.40</td>
<td>Spindles</td>
<td>15-667</td>
</tr>
<tr>
<td>15.2.41</td>
<td>Polynomial values for synchronized actions</td>
<td>15-670</td>
</tr>
<tr>
<td>15.2.42</td>
<td>Channel states</td>
<td>15-672</td>
</tr>
<tr>
<td>15.2.43</td>
<td>Measurement</td>
<td>15-673</td>
</tr>
<tr>
<td>15.2.44</td>
<td>Positions</td>
<td>15-677</td>
</tr>
<tr>
<td>15.2.45</td>
<td>Indexing axes</td>
<td>15-679</td>
</tr>
<tr>
<td>15.2.46</td>
<td>Encoder values</td>
<td>15-679</td>
</tr>
<tr>
<td>15.2.47</td>
<td>Axial measurement</td>
<td>15-680</td>
</tr>
<tr>
<td>15.2.48</td>
<td>Offsets</td>
<td>15-681</td>
</tr>
<tr>
<td>15.2.49</td>
<td>Axial paths</td>
<td>15-684</td>
</tr>
<tr>
<td>15.2.50</td>
<td>Oscillation</td>
<td>15-685</td>
</tr>
<tr>
<td>15.2.51</td>
<td>Axial velocities</td>
<td>15-685</td>
</tr>
<tr>
<td>15.2.52</td>
<td>Drive data</td>
<td>15-687</td>
</tr>
<tr>
<td>15.2.53</td>
<td>Axis statuses</td>
<td>15-688</td>
</tr>
<tr>
<td>15.2.54</td>
<td>Master/slave links</td>
<td>15-689</td>
</tr>
<tr>
<td>15.2.55</td>
<td>Travel to fixed stop</td>
<td>15-690</td>
</tr>
<tr>
<td>15.2.56</td>
<td>Electronic gear</td>
<td>15-691</td>
</tr>
<tr>
<td>15.2.57</td>
<td>Leading value coupling</td>
<td>15-692</td>
</tr>
</tbody>
</table>
15.2.58 Synchronized spindle ................................................................. 15-693
15.2.59 Safety Integrated ........................................................................ 15-696

Appendix

A Index ........................................................................................................... A-702
B Commands, Identifiers ............................................................................... A-719
Preface

Overview of documentation

The SINUMERIK documentation is organized in three parts:
- General Documentation
- User Documentation
- Manufacturer/Service Documentation

Target group

This documentation is intended for the programmer. It provides detailed information for programming the SINUMERIK 840D/840Di/810D.

Standard scope

The Programming Guide describes the functionality included in the standard scope. Extensions or changes made by the machine tool manufacturer are documented by the machine tool manufacturer.

You can obtain more detailed information on publications about SINUMERIK 840D/840Di/810D or publications that apply to all the SINUMERIK controls (e.g. universal interface, measurement cycles, etc.), from your Siemens branch.

Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.
Validity
This Programming Guide is valid for the following controls:
SINUMERIK 840D  SW6
SINUMERIK 840DE (export version)  SW6
SINUMERIK 840Di  SW2
SINUMERIK 840DiE (export version)  SW2
SINUMERIK 810D  SW3
SINUMERIK 810DE (export version)  SW3
with operator panel fronts OP 010, OP 010C, OP 010S, OP 12 or OP 15 (PCU 20 or PCU 50)

SINUMERIK 840D powerline
From 09.2001, the
• SINUMERIK 840D powerline and the
• SINUMERIK 840DE powerline
will be available with improved performance. A list of the available powerline modules can be found in the
Hardware Reference Manual
/PHD/ in Section 1.1

SINUMERIK 810D powerline
From 12.2001, the
• SINUMERIK 810D powerline and the
• SINUMERIK 810DE powerline
will be available with improved performance. A list of the available powerline modules can be found in the
Hardware Reference Manual
/PHC/ in Section 1.1
Hotline

Should you have any questions, please consult the following Hotline:
A&D Technical Support Tel.: ++49-(0)180-5050-222
Fax: ++49-(0)180-5050-223
E-mail: adsupport@siemens.com

If you have any questions about the documentation (suggestions, corrections) please send a fax to the following fax address, or e-mail us:
Fax: ++49-(0)0131-98-2176
E-mail: motioncontrol.docu@erlf.siemens.de
Fax form: see the feedback page at the back of this document.

Internet address
http://www.ad.siemens.de/sinumerik

Export version

The following functions are not available in the export version:

<table>
<thead>
<tr>
<th>Function</th>
<th>810DE</th>
<th>840DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machining package for 5 axes</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Transformation package handling (5 axes)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Multiple axes interpolation (&gt; 4 axes)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Helix interpolation 2D+6</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Synchronized actions stage 2</td>
<td>–</td>
<td>O¹)</td>
</tr>
<tr>
<td>Measurement stage 2</td>
<td>–</td>
<td>O¹)</td>
</tr>
<tr>
<td>Adaptive control</td>
<td>O¹)</td>
<td>O¹)</td>
</tr>
<tr>
<td>Continuous dressing</td>
<td>O¹)</td>
<td>O¹)</td>
</tr>
<tr>
<td>Use of the compile cycles (OEM)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Multidimensional sag compensation</td>
<td>O¹)</td>
<td>O¹)</td>
</tr>
</tbody>
</table>

¹) Limited functionality
Structure of the descriptions

All cycles and programming options have been described – where appropriate and possible – according to the same internal structure. The organization into different information levels allows you to find the information you need quickly.

1. At a glance

If you want to look up a seldom used command or the meaning of a parameter, you can see at a glance how to program the function together with an explanation of the commands and parameters.

This information is always presented at the start of the page.

Note:
To keep this documentation as compact as possible, it is not always possible to list all the types of representation available in the programming language for the individual commands and parameters. The commands are therefore always programmed in the context most frequently used in the workshop.
2. Detailed explanations

The theory part contains detailed information on the following:

- What is the purpose of the command?
- What is the effect of the command?
- What is the sequence of command?
- What effect do the parameters have?
- What else has to be taken into account?

The theory parts are suitable primarily as a guide for NC beginners. Work through the manual carefully at least once to gain an overview of the performance scope and capabilities of your SINUMERIK control.

3. From theory to practice

The programming example shows you how to apply the commands in the program.

You will find an application example for practically all the commands after the theory part.
Explanation of the symbols

Sequence of operations

Explanation

Function

Parameters

Programming example

Programming

Additional notes

Cross-references to other documentation and sections

Important information and safety notices
For your information

Your SINUMERIK 840D/840Di/810D is state of the art and is manufactured in accordance with recognized safety regulations, standards and specifications.

Additional devices

SIEMENS offers special add-on equipment, products and system configurations for the focused expansion of SIEMENS controls in your field of application.

Personnel

Only specially trained, authorized and experienced personnel should be allowed to work on the control. This applies at all times, even for short periods.

It is necessary to clearly define the respective responsibilities of the personnel for setting up, operation and maintenance; it is necessary to supervise the compliance thereof.

Actions

It must be ascertained that the Instruction Manuals have been read and understood by the persons working on the control before installation and start-up of the control. In addition, operation must be conducted under constant supervision regarding the overall technical state (faults and damages visible from outside, as well as changes in operation behavior) of the control.
Service

Only qualified personnel specifically trained for this purpose should be allowed to perform repairs, and only in accordance with the contents of the maintenance guides. Hereby, all established safety regulations have to be complied with.

Note

The following are considered not compliant with the usage to the intended purposes and are therefore excluded from all liability of the manufacturer:

Every usage not complying with or going beyond the abovementioned points.

If the control is not operated in a technically faultless state, if proper safety precautions are not taken, or if the instructions in the Instruction Manual are not complied with.

If faults which could influence safety of operation are not remedied before installation and start-up of the control.

Each change, jumpering or shut-down of devices on the control which serve for proper functioning, universal usage and active and passive safety.

Unforeseen dangers may result in:

- personal injury and death,
- damage to the control, machine and other property of the company and operator.
The following notes used in the documentation have a special significance:

**Notes**
This symbol always appears in the documentation if secondary information is given and there is an important fact to be considered.

In this documentation, you will find the symbol shown with reference to an ordering data option. The function described can only be run if the control includes the designated option.

**Warnings**
The following warnings, of graduated significance, are used in the publication.

**Danger**
Indicates an imminently hazardous situation which, if not avoided, **will** result in death or serious injury or in substantial property damage.

**Notice**
Indicates a potentially hazardous situation which, if not avoided, **could** result in death or serious injury or in substantial property damage.

**Caution**
Used with the safety alert symbol indicates a potentially hazardous situation which, if not avoided, **may** result in minor or moderate injury or in property damage.

**Caution**
Used without safety alert symbol indicates a potentially hazardous situation which, if not avoided, **may** result in property damage.
Notice
Used without the safety alert symbol indicates a potential situation which, if not avoided, may result in an undesirable result or state.
Flexible NC Programming

1.1 Variable and arithmetic parameters ................................................................. 1-26
1.2 Variable definition .............................................................................................. 1-29
1.3 Array definition ................................................................................................. 1-34
1.4 Indirect programming ....................................................................................... 1-40
1.5 Assignments ...................................................................................................... 1-45
1.6 Arithmetic operations and functions ................................................................... 1-46
1.7 Comparison and logic operators ....................................................................... 1-48
1.8 Priority of operators ......................................................................................... 1-53
1.9 Possible type conversions .................................................................................. 1-54
1.10 String operations ............................................................................................. 1-55
1.10.1 Type conversion ......................................................................................... 1-56
1.10.2 Concatenation of strings .............................................................................. 1-58
1.10.3 Conversion to lower/upper case .................................................................. 1-59
1.10.4 Length of the string .................................................................................... 1-60
1.10.5 Search for character/string in a string ......................................................... 1-60
1.10.6 Selection of a substring ............................................................................. 1-62
1.10.7 Selection of a single character ................................................................. 1-63
1.11 CASE instruction ............................................................................................ 1-65
1.12 Control structures ......................................................................................... 1-67
1.13 Program coordination .................................................................................... 1-72
1.14 Interrupt routine ............................................................................................ 1-77
1.15 Axis transfer, spindle transfer ........................................................................ 1-85
1.16 NEWCONF: Setting machine data active (SW 4.3 and higher) ......................... 1-90
1.17 WRITE: Write file (SW 4.3 and higher) .......................................................... 1-91
1.18 DELETE: Delete file (SW 4.3 and higher) ....................................................... 1-93
1.19 READ: Read lines in file (SW 5.2 and higher) ............................................... 1-94
1.20 ISFILE: File available in user memory NCK (SW 5.2 and higher) ................. 1-97
1.21 CHECKSUM: Creation of a checksum over an array (SW 5.2 and higher) .... 1-98
1.1 Variable and arithmetic parameters

Function

Using variables in place of constant values makes a program more flexible. You can respond to signals such as measured values or, by storing setpoints in the variables, you can use the same program for different geometries.

With variable calculation and jump instructions a skilled programmer is able to create a very flexible program archive and save a lot of programming work.

Variable classes

The controller uses 3 classes of variable:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined variable</td>
<td>Name and type of variable defined by the user, e.g. arithmetic parameter.</td>
</tr>
<tr>
<td>Arithmetic parameter</td>
<td>Special, predefined arithmetic variable whose address is R plus a number. The predefined arithmetic variables are of the REAL type.</td>
</tr>
<tr>
<td>System variable</td>
<td>Variable provided by the controller that can be processed in the program (write, read). System variables provide access to zero offsets, tool offsets, actual values, measured values on the axes, control states, etc. (See Appendix for the meaning of the system variables)</td>
</tr>
</tbody>
</table>
11.02 Flexible NC Programming

1.1 Variable and arithmetic parameters

Variable types

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Integers with sign</td>
<td>±(2^{31} - 1)</td>
</tr>
<tr>
<td>REAL</td>
<td>Real numbers (fractions with decimal point, LONG REAL according to IEEE)</td>
<td>±(10^{-300} … 10^{+300})</td>
</tr>
<tr>
<td>BOOL</td>
<td>Boolean values: TRUE (1) and FALSE (0)</td>
<td>1, 0</td>
</tr>
<tr>
<td>CHAR</td>
<td>1 ASCII character specified by the code</td>
<td>0 … 255</td>
</tr>
<tr>
<td>STRING</td>
<td>Character string, number of characters in [...], Max. 200 characters</td>
<td>Sequence of values with 0 … 255</td>
</tr>
<tr>
<td>AXIS</td>
<td>Axis names (axis addresses) only</td>
<td>All axis identifiers and spindles in the channel</td>
</tr>
<tr>
<td>FRAME</td>
<td>Geometric data for translation, rotation, scaling, mirroring, see Chapter 4.</td>
<td></td>
</tr>
</tbody>
</table>

Arithmetic variable

Address R provides 100 arithmetic variables of type REAL by default.

The exact number of arithmetic variables (up to 1000) is defined in machine data.

Example: \( R10 = 5 \)

System variable

The controller provides system variables that can be contained and processed in all running programs.

System variable provide machine and controller states. Some of the system variables cannot be assigned values.
1.1 Variable and arithmetic parameters

Special identifiers of system variables always begin with a "$" sign followed by the specific names.

<table>
<thead>
<tr>
<th>1st letter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M</td>
<td>Machine data</td>
</tr>
<tr>
<td>$S</td>
<td>Setting data</td>
</tr>
<tr>
<td>$T</td>
<td>Tool management data</td>
</tr>
<tr>
<td>$P</td>
<td>Programmed values</td>
</tr>
<tr>
<td>$A</td>
<td>Current values</td>
</tr>
<tr>
<td>$V</td>
<td>Service data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2nd letter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>NCK global</td>
</tr>
<tr>
<td>C</td>
<td>Channel-specific</td>
</tr>
<tr>
<td>A</td>
<td>Axis-specific</td>
</tr>
</tbody>
</table>

Example: $AA.IM
Means: Current axis-specific value in the machine coordinate system.
1.2 Variable definition

User-defined variables
The programmer can define and assign values to variables in addition to using predefined variables. Local variables (LUD) are only valid in the program where they are defined. Global variables (GUD) are valid in all programs.

SW 4.4 and higher:
Machine data are used to redefine the local user variables (LUD) defined in the main program as program-global user variables (PUD).

Machine manufacturer
See machine manufacturer's specifications.

If they are defined in the main program, they will also be valid at all levels of the subprograms called. They are created with parts program start and deleted with parts program end or reset.

Example:
$MN_LUD_EXTENDED_SCOPE=1

PROC MAIN ;Main program
DEF INT VAR1 ;PUD definition
...
SUB2 ;Subprogram call
...
M30

PROC SUB2 ;Subprogram SUB2
DEF INT VAR2 ;LUD DEFINITION
...
IF (VAR1==1) ;Read PUD
  VAR1=VAR1+1 ;Read & write PUD
  VAR2=1 ;Write LUD
ENDIF
SUB3 ;Subprogram call
...
M17
PROC SUB3 ;Subprogram SUB3
**Variable definition**

If machine data `$MN_LUD_EXTENDED_SCOPE` is set, it is not possible to define a variable with the same name in the main and subprograms.

**Variable names**

A variable name consists of up to 31 characters. The first two characters must be a letter or an underscore.

The "$" sign can not be used for user-defined variables because it is used for system variables.

**Programming**

DEF INT name
or DEF INT name=value

DEF REAL name
or DEF REAL name1,name2=3,name4
or DEF REAL name[array_index1,array_index2]

DEF BOOL name

DEF CHAR name
or DEF CHAR name[array_index]=("A","B",...)

DEF STRING[string_length] name

DEF AXIS name
or DEF AXIS name[array_index]

DEF FRAME name
If a variable is not assigned a value on definition, the system sets zero as the default.

Variables must be defined at the beginning of the program before they are used. The definition must be made in a separate block; only one variable type can be defined per block.

**Explanation**

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>Variable type integer, i.e. whole number</td>
</tr>
<tr>
<td>REAL</td>
<td>Variable type real, i.e. factional number with decimal point</td>
</tr>
<tr>
<td>BOOL</td>
<td>Variable type Boolean, i.e. 1 or 0 (TRUE or FALSE)</td>
</tr>
<tr>
<td>CHAR</td>
<td>Variable type char, i.e. ASCII-coded character (0 to 255)</td>
</tr>
<tr>
<td>STRING</td>
<td>Variable type string, i.e. sequence of char</td>
</tr>
<tr>
<td>AXIS</td>
<td>Variable type axis, i.e. axis addresses and spindles</td>
</tr>
<tr>
<td>FRAME</td>
<td>Variable type frame, i.e. geometric data</td>
</tr>
</tbody>
</table>

**Programming examples**

**Variable type INT**

- **DEF INT NUMBER**
  - This creates a variable of type integer with the name NUMBER.
  - The system initializes the variable with zero.

- **DEF INT NUMBER=7**
  - This creates a variable of type integer with the name NUMBER.
  - The system initializes the variable with zero.

**Variable type REAL**

- **DEF REAL DEPTH**
  - This creates a variable of type real with the name DEPTH.
  - System initializes with zero (0.0).

- **DEF REAL DEPTH=6.25**
  - This creates a variable of type real with the name DEPTH. The variable is initialized with 6.25.

- **DEF REAL DEPTH=3.1,LENGTH=2,NUMBER**
  - More than one variable can be defined in a line.
Variable type **BOOL**

DEF BOOL IF_TOO_MUCH

This creates a variable of type BOOL with the name IF_TOO_MUCH. System initializes with zero (FALSE).

DEF BOOL IF_TOO_MUCH=1 or DEF BOOL IF_TOO_MUCH=TRUE or DEF BOOL WENN_ZUVIEL=FALSE

This creates a variable of type BOOL with the name IF_TOO_MUCH.

---

Variable type **CHAR**

DEF CHAR GUSTAV_1=65

A code value for the corresponding ASCII character or the ASCII character itself can be assigned to a variable of type CHAR (code value 65 corresponds to letter "A").

DEF CHAR GUSTAV_1="A"

---

Variable type **STRING**

DEF STRING[6] MUSTER_1="BEGIN"

Variables of type string can contain a string (sequence of characters). The maximum number of characters is enclosed in square brackets after the variable type.

---

Variable type **AXIS**

DEF AXIS AXIS_NAME=(X1)

Variable of type AXIS are called AXIS_NAME and contain the axis identifier of a channel – here X1. (Axis names with an extended address are in parentheses.)

---

Variable type **FRAME**

DEF FRAME BEVEL_1

Variables of type FRAME have names like BEVEL_1.
A variable of type AXIS can contain an axis identifier and a spindle identifier of a channel. 

Note: 
Axis names with an extended address must be in parentheses.

Example of programming with program-local variables

```plaintext
DEF INT COUNT
LOOP: G0 X… ;Loop
COUNT=COUNT+1
IF COUNT<50 GOTOB LOOP
M30
```

Programming example

Query of existing geometry axes

```plaintext
DEF AXIS ABSCISSA; ;1. geometry axis
IF ISAXIS(1) == FALSE GOTOF CONTINUE
ABSCISSA = $P_AXN1
...
CONTINUE:
```

Indirect spindle programming

```plaintext
DEF AXIS SPINDLE
SPINDLE=(S1)
OVRA[SPINDLE]=80 ;Spindle override = 80%
SPINDLE=(S3)
...
```
1.3 Array definition

Programming

DEF CHAR NAME[n,m]
DEF INT NAME[n,m]
DEF REAL NAME[n,m]
DEF AXIS NAME[n,m]
DEF FRAME NAME[n,m]
DEF STRING[string_length] NAME[m]
DEF BOOL[n,m]

Explanation

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Memory requirement per array element</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>1 byte</td>
</tr>
<tr>
<td>CHAR</td>
<td>1 byte</td>
</tr>
<tr>
<td>INT</td>
<td>4 bytes</td>
</tr>
<tr>
<td>REAL</td>
<td>8 bytes</td>
</tr>
<tr>
<td>STRING</td>
<td>String length + 1</td>
</tr>
<tr>
<td>FRAME</td>
<td>~ 400 bytes, depending on number of axes</td>
</tr>
<tr>
<td>AXIS</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

The same memory size applies to type BOOL as to type CHAR.

Up to SW3:
The maximum size of an array is set via machine data.

Machine manufacturer

See machine manufacturer’s specifications

The maximum array size determines the size of the memory blocks in which the variable memory is managed. It should not be set higher than actually required.

Standard: 812 bytes
If not large arrays are defined, select: 256 bytes.
11.02 Flexible NC Programming

1.3 Array definition

SW 4 and higher:
An array can be larger than a memory block. The MD value for block size should be set such that arrays are fragmented only in exceptional cases. Default: 256 bytes
Memory requirement per element: see above

Example:
Global user data must contain PLC machine data for switching the controller on/off (definition of BOOL arrays).

Additional notes
Arrays with up to 2 dimensions can be defined.

Arrays with variables of type STRING can only be 1-dimensional. The string length is specified after the data type String.

Array index
Elements of an array are accessed via the array index. The array elements can either be read or assigned values using this array index.

The first array element starts with index [0,0]; for example, for array size [3,4] the maximum possible array index is [2,3].
In the above example, the values have been initialized to double as the index of the array element. In order to illustrate the sequence of the individual array elements.

Initialization of arrays

The array elements can be initialized during program run or in the array definition.

In 2-dimensional arrays, the right array index is increment first.

Initialization with value lists, SET

1. Initializing in the array definition

DEF Type VARIABLE = SET(VALUE)
DEF Type ARRAY[n,m] = SET(VALUE, value, ...)

Or:
DEF Type VARIABLE = Value
DEF Type ARRAY[n,m] = (value, value, ...)

- As many array elements are assigned as initialization values are programmed.
- Array elements without values (gaps in the value list) are automatically initialized to 0.
- For variables of type AXIS, gaps in the value list are not permitted.
- Programming more values than exist in the remaining array elements triggers an alarm.

Example:
DEF REAL ARRAY[2,3]=(10, 20, 30, 40)

SET is optional in the array definition.
2. Initializing during the program run

ARRAY\[n,m\]= SET(value, value, value, ...)
ARRAY\[n,m\]= SET(expression, expression, expression, ...)

- Initialization is the same as in array definition.
- Expressions are possible values in this case too.
- Initialization starts at the programmed array indexes. Values can also be assigned selectively to subarrays.

Example:
Assignment of expressions
DEF INT ARRAY\[5, 5\]
ARRAY\[0,0\] = SET(1, 2, 3, 4, 5)
ARRAY\[2,3\] = SET(VARIABLE, 4*5.6)

The axis index of axis variables is not traversed:
Example:
Initialization in one line
$MA\_AX\_VELO\_LIMIT[1, AX1] = SET(1.1, 2.2, 3.3)$

Is equivalent to:
$MA\_AX\_VELO\_LIMIT[1,AX1] = 1.1$
$MA\_AX\_VELO\_LIMIT[2,AX1] = 2.2$
$MA\_AX\_VELO\_LIMIT[3,AX1] = 3.3$
### Initialization with the same values, REP

**1. Initializing in the array definition**

\[
\text{DEF Type ARRAY}[n,m] = \text{REP}(\text{value})
\]

All array elements are assigned the same value (constant).

Variables of type FRAME cannot be initialized.

**Example:**

\[
\text{DEF REAL ARRAY5}[10,3] = \text{REP}(9.9)
\]

**2. Initializing during the program run**

\[
\text{ARRAY}[n,m] = \text{REP}(\text{value})
\]

- Expressions are possible values in this case too.
- All array elements are initialized to the same value.
- Initialization starts at the programmed array indexes. Values can also be assigned selectively to subarrays.

Variables of type FRAME are permissible and can initialized very simply in this way.

**Example:**

Initialization of all elements with one value

\[
\text{DEF FRAME FRM}[10] \text{ FRM}[5] = \text{REP} (\text{CTRANS}(X,5))
\]
Programming example

Initialization of complete variable arrays.
The current assignment is shown in the drawing.

N10 DEF REAL FELD1[10,3] = SET(0, 0, 0, 10, 11, 12, 20, 20, 20, 30, 30, 30, 40, 40, 40,)
N20 FELD1[0,0] = REP(100)
N30 FELD1[5,0] = REP(-100)
N40 FELD1[0,0] = SET(0, 1, 2, -10, -11, -12, -20, -20, -20, -30, , , , -40, -40, -50, -60, -70)
N50 FELD1[8,1] = SET(8.1, 8.2, 9.0, 9.1, 9.2)

Array index

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The array elements [5.0] to [9.2] have been initialized with the default value (0.0).

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

The array elements [3.1] to [4.0] have been initialized with the different values.

The array elements [6.0] to [8.0] have not been changed.
1.4 Indirect programming

Indirect programming permits general-purpose use of programs. The extended address (index) is substituted by a variable of suitable type.

All addresses are parameterizable except:
- N – Block number
- G – G command
- L – Subprogram

Indirect programming is not possible for settable addresses. Example: X[1] in place of X1 is not permissible.

Programming

ADDRESS[INDEX]

Programming examples

Spindle

<table>
<thead>
<tr>
<th>Programming</th>
<th>Direct programming</th>
<th>Indirect programming:</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1=300</td>
<td></td>
<td>Speed 300rpm for the spindle whose number is stored in the SPINU variable (in this example 1).</td>
</tr>
<tr>
<td>DEF INT SPINU=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S[SPINU]=300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Feed

<table>
<thead>
<tr>
<th>Programming</th>
<th>Direct programming</th>
<th>Indirect programming:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA[U]=300</td>
<td></td>
<td>Feedrate for positioning axis whose address name is stored in the variable of type AXIS with the variable name AXVAR2.</td>
</tr>
<tr>
<td>DEF AXIS AXVAR2=U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FA[AXVAR2]=300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Measured value

<table>
<thead>
<tr>
<th>Programming</th>
<th>Direct programming</th>
<th>Indirect programming:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_MM[X]</td>
<td></td>
<td>Measured value in machine coordinates for the axis whose name is stored in variable AXVAR3.</td>
</tr>
<tr>
<td>DEF AXIS AXVAR3=X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_MM[AXVAR3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.02 Flexible NC Programming

1.4 Indirect programming

Array element

<table>
<thead>
<tr>
<th>Direct programming</th>
<th>Indirect programming:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF INT FELD1[4,5]</td>
<td>Array dimensions must be stated as constant values.</td>
</tr>
<tr>
<td>DEFINE DIM1 AS 4</td>
<td></td>
</tr>
<tr>
<td>DEFINE DIM2 AS 5</td>
<td></td>
</tr>
<tr>
<td>DEF INT ARRAY[DIM1,DIM2]</td>
<td></td>
</tr>
<tr>
<td>ARRAY[DIM1-1,DIM2-1]=5</td>
<td></td>
</tr>
</tbody>
</table>

Axis assignment with axis variables

<table>
<thead>
<tr>
<th>Direct programming</th>
<th>Indirect programming:</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1=100 X2=200</td>
<td></td>
</tr>
<tr>
<td>DEF AXIS AXVAR1 AXVAR2</td>
<td>Definition of the variables</td>
</tr>
<tr>
<td>AXVAR1=(X1) AXVAR2=(X2)</td>
<td>Assignment of the axis names, traversal of axes that are stored in the variables to 100 or 200.</td>
</tr>
<tr>
<td>AX[AXVAR1]=100 AX[AXVAR2]=200</td>
<td></td>
</tr>
</tbody>
</table>

Interpolation parameters with axis variables

<table>
<thead>
<tr>
<th>Direct programming</th>
<th>Indirect programming:</th>
</tr>
</thead>
<tbody>
<tr>
<td>G2 X100 I20</td>
<td></td>
</tr>
<tr>
<td>DEF AXIS AXVAR1=X</td>
<td>Definition and assignment of the axis name</td>
</tr>
<tr>
<td>G2 X100 IP[AXVAR1]=20</td>
<td>Indirect programming of the center</td>
</tr>
</tbody>
</table>

Indirect subprogram call

| Call of the program whose number is in R10 |
| CALL "L" << R10 |

Additional notes

R parameters can also be considered 1-dimensional arrays with abbreviated notation (R10 is equivalent to R([10])).
1.4 Indirect programming

Indirect G code programming from SW 5
G[<Group index>] = <integer/real variable>

Indirect programming of G codes using variables for effective cycle programming

Meaning of the parameters

<table>
<thead>
<tr>
<th>&lt;Group index&gt;</th>
<th>Integer constants with which the G code group is selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;integer/real variable&gt;</td>
<td>Variable of the integer or real type with which the G code number is selected.</td>
</tr>
</tbody>
</table>

Function

Indirect G code programming (SW 5 and higher)
The indirect programming of G codes using variables facilitates effective cycle programming.
Two parameters
- G code groups integer constant
- G code numbers variable of the integer/real type
are available for this.

Valid G code groups
Only modal G code groups can be programmed indirectly.
Non-modal G code groups are rejected by alarm 12470.

Valid G code numbers
Arithmetic functions are not legal in indirect G code programming.
The G code number must be stored in a variable of the integer or real type. Invalid G code numbers are rejected by alarm 12475.
If it is necessary to calculate the G code number, this must be done in a separate parts program line before the indirect G code programming.

Additional notes
All the valid G codes are shown in the PG, in the "List of G functions/preparatory functions" section in various groups.
See /PG/ Fundamentals Programming Guide, "Tables"
Programming example

Indirect G code programming

; Settable zero offset G code group 8
N1010 DEF INT INT_VAR
N1020 INT_VAR = 2
...
N1090 G[8] = INT_VAR G1 X0 Y0 ; G54
N1100 INT_VAR = INT_VAR + 1 ; G code calculation
N1110 G[8] = INT_VAR G1 X0 Y0 ; G55

; Plane selection G code group 6
N2010 R10 = $P_GG[6] ; Read G code for current plane
...
N2090 G[6] = R10 ; G17 – G19
1.4 Indirect programming

Run string as parts program line

EXECSTRING (<string variable>)

Command EXECSTRING runs a parts program line indirectly.

Meaning of the parameters

| <string variable> Parameter of type string is transferred with EXECSTRING |

Function

EXECSTRING (from SW 6.4)
Parts program command EXECSTRING transfers a string as a parameter that already contains the parts program line to run.

Additional notes
All parts program constructions that can be programmed in a parts program can be output. That excludes PROC and DEF instructions and all use of INI and DEF files.

Programming example

Indirect parts program line

<table>
<thead>
<tr>
<th>String variable to be included in parts program line</th>
</tr>
</thead>
<tbody>
<tr>
<td>String variable to be included in parts program line</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run parts program line &quot;M7 M4711&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run parts program line &quot;M3 M4711&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Run parts program line &quot;M3 M4711&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run parts program line &quot;M3 M4711&quot;</td>
</tr>
</tbody>
</table>
1.5 Assignments

Values of a suitable type can be assigned to the variables/arithmetic parameters in the program.

Assignments to axis addresses (traversing instructions) always require a separate block to variable assignments. Assignment to axis addresses (traverse instructions) must be in a separate block from the variable assignments.

Programming example

<table>
<thead>
<tr>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1=10.518 R2=4 VARI1=45</td>
</tr>
<tr>
<td>X=47.11 Y=R2</td>
</tr>
<tr>
<td>R1=R3 VARI1=R4</td>
</tr>
<tr>
<td>R4=-R5 R7=-VARI8</td>
</tr>
</tbody>
</table>

Assignment of a numeric value
Assignment of a suitable type variable
Assignment with opposite sign (only permitted for types INT and REAL)

Assignment to string variable

CHARs and STRINGs distinguish between upper and lower case.
If you want to include an ' or " in the string, put it in single quotes '…'.

Example:
MSG("Viene lavorata l' 'ultima figura")
displays the text 'Viene lavorata l'ultima figura' on the screen.

The string can contain non-displayable characters if they are specified as binary or hexadecimal constants.
### 1.6 Arithmetic operations and functions

The arithmetic functions are primarily for R parameters and variables (or constants and functions) of type REAL. The types INT and CHAR are also permitted.

Use of arithmetic operations requires conventional mathematical notation. Priorities for execution are indicated by parentheses. Angles are specified for trigonometry functions and their inverse functions (right angle = 90°).

#### Operators/arithmetic functions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>NOTICE: (Type INT)/(Type INT)=(Type REAL); Example: 3/4 = 0.75</td>
<td></td>
</tr>
<tr>
<td>DIV</td>
<td>Division, for variable type INT and REAL</td>
</tr>
<tr>
<td>NOTICE: (Type INT)DIV(Type INT)=(Type INT); Example: 3 DIV 4 = 0</td>
<td></td>
</tr>
<tr>
<td>MOD</td>
<td>Modulo division (INT or REAL) produces remainder of INT division, e.g. 3 MOD 4 = 3</td>
</tr>
<tr>
<td>::</td>
<td>Chain operator (for FRAME variables)</td>
</tr>
<tr>
<td>Sin()</td>
<td>Sine</td>
</tr>
<tr>
<td>COS()</td>
<td>Cosine</td>
</tr>
<tr>
<td>TAN()</td>
<td>Tangent</td>
</tr>
<tr>
<td>ASIN()</td>
<td>Arcsine</td>
</tr>
<tr>
<td>ACOS()</td>
<td>Arccosine</td>
</tr>
<tr>
<td>ATAN2(,)</td>
<td>Arctangent2</td>
</tr>
<tr>
<td>SQRT()</td>
<td>Square root</td>
</tr>
<tr>
<td>ABS()</td>
<td>Absolute number</td>
</tr>
<tr>
<td>POT()</td>
<td>2nd power (square)</td>
</tr>
<tr>
<td>TRUNC()</td>
<td>Truncate to integer</td>
</tr>
<tr>
<td>ROUND()</td>
<td>Round to integer</td>
</tr>
<tr>
<td>LN()</td>
<td>Natural logarithm</td>
</tr>
<tr>
<td>EXP()</td>
<td>Exponential function</td>
</tr>
<tr>
<td>CTRANS()</td>
<td>Translation</td>
</tr>
<tr>
<td>CROT()</td>
<td>Rotation</td>
</tr>
</tbody>
</table>
11.02 Flexible NC Programming

1.6 Arithmetic operations and functions

Programming examples

CSCALE() Scaling
CMIRROR() Mirroring

Arithmetic function ATAN2( , )
The function calculates the angle of the total vector from two mutually orthogonal vectors. The result is in one of four quadrants (−180 ° ≤ 0 ° < +180 °). The angular reference is always based on the 2nd value in the positive direction.

\[
R1 = R1 + 1
\]
\[
R1 = R2 + R3 \quad R4 = R5 - R6 \quad R7 = R8 \times R9
\]
\[
R10 = R11 / R12 \quad R13 = \sin(25.3)
\]
\[
R14 = R1 \times R2 + R3
\]
\[
R14 = (R1 + R2) \times R3
\]
\[
R15 = \sqrt{\text{POT}(R1) + \text{POT}(R2)}
\]
\[
\text{RESFRAME} = \text{FRAME1:FRAME2}
\]
\[
\text{FRAME3} = \text{TRANS}(...) : \text{CROT}(...) \quad \text{The concatenation operator links frames to form a resulting frame or assigns values to frame components}
\]

\[
R3 = \text{ATAN2}(30.5, 80.1)
\]
\[
R3 = \text{ATAN2}(30, -80)
\]

Angle=20.8455°
Angle=159.444°
1.7 Comparison and logic operators

Comparison operators

The comparison operations are applicable to variables of type CHAR, INT, REAL, and BOOL. The code value is compared with the CHAR type.

For types STRING, AXIS, and FRAME, the following are possible: == and <>.

The result of comparison operations is always of type BOOL.

Comparison operations can be used, for example, to formulate a jump condition. Complex expressions can also be compared.

Meaning of comparison operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equal to</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
</tbody>
</table>

Programming example

IF R10>=100 GOTOF DEST
or
R11=R10>=100
IF R11 GOTOF DEST

The result of the R10>=100 comparison is first buffered in R11.
Precision correction on comparison errors

**TRUNC (R1*1000)**

The TRUNC command truncates the operand multiplied by a precision factor.

**Function**

**Settable precision for comparison commands**

Program data of type REAL are displayed internally with 64 bits in IEEE format. This display format can cause decimal numbers to be displayed imprecisely and lead to unexpected results when compared with the ideally calculated values.

**Relative equality**

To prevent the imprecision caused by the display format from interfering with program flow, the comparison commands do not check for absolute equality but for relative equality.

**SW 6.3 and lower**

Relative equality considered $10^{-12}$ for

- Equality (==)
- Inequality (<>)
- Greater than or equal to (>=)
- Less than or equal to (<=)
- Greater/less than (> <) with absolute equality

**SW 6.4 and higher**

Relative equality considered $10^{-12}$ for

- Greater than (>
- Less than (<

**Programming notes**

Comparisons with data of type REAL are subject to a certain imprecision for the above reasons. If deviations are unacceptable, use INTEGER calculation by multiplying the operands by a precision factor and then truncating with TRUNC.
### Synchronized actions
The response described for the comparison commands also applies to synchronized actions.

**Compatibility**
For compatibility reasons, the check for relative equality with (>) and (<) can be deactivated by setting MD 10280: PROG_FUNCTION_MASK Bit0 = 1.

### Programming examples

#### Precision issues

<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N40</td>
<td>R1=61.01 R2=61.02 R3=0.01</td>
<td>Assignment of initial values</td>
</tr>
<tr>
<td>N41</td>
<td>IF ABS(R2-R1) &gt; R3 GOTOF ERROR</td>
<td>Jump executed (SW 6.3 and lower)</td>
</tr>
<tr>
<td>N42</td>
<td>M30</td>
<td>End of program</td>
</tr>
<tr>
<td>N43</td>
<td>ERROR: SETAL(66000)</td>
<td></td>
</tr>
</tbody>
</table>

R1=61.01 R2=61.02 R3=0.01 Assignment of initial values

R11=TRUNC(R1*1000) R12=TRUNC(R2*1000) R13=TRUNC(R3*1000) Precision correction

IF ABS(R12-R11) > R13 GOTOF ERROR Jump not executed

M30 End of program

ERROR: SETAL(66000)

#### Calculate and evaluate quotient of both operands

<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N40</td>
<td>R1=61.01 R2=61.02 R3=0.01</td>
<td>Assignment of initial values</td>
</tr>
<tr>
<td>N41</td>
<td>IF ABS((R2-R1)/R3)-1) &gt; 10EX-5 GOTOF ERROR</td>
<td>Jump not executed</td>
</tr>
<tr>
<td>N42</td>
<td>M30</td>
<td>End of program</td>
</tr>
<tr>
<td>N43</td>
<td>ERROR: SETAL(66000)</td>
<td></td>
</tr>
</tbody>
</table>

R1=61.01 R2=61.02 R3=0.01 Assignment of initial values

IF ABS((R2-R1)/R3)-1) > 10EX-5 GOTOF ERROR Jump not executed

M30 End of program

ERROR: SETAL(66000)
Logic operators

Logic operators are used to link truth values. AND, OR, NOT, and XOR can only be applied to variables of type BOOL. However, they can also be applied to data types CHAR, INT, and REAL by implicit type conversion.

Spaces must be left between BOOLEAN operands and operators.

For the logic (Boolean) operations, the following applies to data types BOOL, CHAR, INT, and REAL:
- 0 means FALSE
- not equal to 0 means TRUE

Meaning of logic operators

<table>
<thead>
<tr>
<th>Logic operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>AND</td>
</tr>
<tr>
<td>OR</td>
<td>OR</td>
</tr>
<tr>
<td>NOT</td>
<td>Negation</td>
</tr>
<tr>
<td>XOR</td>
<td>Exclusive OR</td>
</tr>
</tbody>
</table>

In arithmetic expressions, the execution order of all the operators can be specified by parentheses, in order to override the normal priority rules.

Programming example

IF (R10<50) AND ($AA_IM[X]>=17.5) GOTOF ZIEL

IF NOT R10 GOTOB START

NOT is only applied to one operand.
Bit logic operators

Logic operations can also be applied to single bits of types CHAR and INT. Type conversion is automatic.

Meaning of bit logic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_AND</td>
<td>Bit AND</td>
</tr>
<tr>
<td>B_OR</td>
<td>Bit OR</td>
</tr>
<tr>
<td>B_NOT</td>
<td>Bit negation</td>
</tr>
<tr>
<td>B_XOR</td>
<td>Bit exclusive OR</td>
</tr>
</tbody>
</table>

The operator B_NOT refers to one operand only, it comes after the operator.

Programming example

```
IF $MC_RESET_MODE_MASK B_AND 'B10000' GOTOF ACT_PLANE
```
1.8 Priority of operators

Priority of the operators

Each operator is assigned a priority. When an expression is evaluated, the operators with the highest priority are always applied first. Where operators have the same priority, the evaluation is from left to right.

In arithmetic expressions, the execution order of all the operators can be specified by parentheses, in order to override the normal priority rules.

Order of operators
(from the highest to lowest priority)

1. NOT, B_NOT
   Negation, bit negation
2. *, /, DIV, MOD
   Multiplication, division
3. +, –
   Addition, subtraction
4. B_AND
   Bit AND
5. B_XOR
   Bit exclusive OR
6. B_OR
   Bit OR
7. AND
   AND
8. XOR
   Exclusive OR
9. OR
   OR
10. <<
    Concatenation of strings, result type STRING
11. ==, <>, >, <, >=, <=
    Comparison operators

Example of IF statement:
If (otto==10) and (anna==20) gotof end

The concatenation operator ":=" for Frames must not be used in the same expression as other operators. A priority level is thus not required for this operator.
1.9 Possible type conversions

Type conversion on assignment

The constant numeric value, the variable, or the expression assigned to a variable must be compatible with the variable type. If this is the case, the type is automatically converted when the value is assigned.

Possible type conversions

<table>
<thead>
<tr>
<th>from</th>
<th>REAL</th>
<th>INT</th>
<th>BOOL</th>
<th>CHAR</th>
<th>STRING</th>
<th>AXIS</th>
<th>FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>INT</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>BOOL</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CHAR</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>–</td>
<td>yes</td>
<td>–</td>
</tr>
<tr>
<td>STRING</td>
<td>–</td>
<td>–</td>
<td>yes</td>
<td>yes</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>AXIS</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>yes</td>
<td>–</td>
</tr>
<tr>
<td>FRAME</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>yes</td>
</tr>
</tbody>
</table>

* During type conversion from REAL to INT, fractional values >= 0.5 are rounded up, others rounded down (cf. ROUND function)

1) Value <> 0 corresponds to TRUE, value == 0 corresponds to FALSE
2) If the value is in the permissible range
3) If only 1 character
4) String length 0 => FALSE, otherwise TRUE

If conversion produces a value greater than the target range, an error message is output.

Additional notes

If mixed types occur in an expression, type conversion is automatic.
1.10 String operations

Overview

Further string manipulations are provided in addition to the conventional operations "Assignment" and "Comparison" described in this section:

Explanation

Type conversion to STRING:

| STRING_ERG = <<bel._Typ" | Result type: STRING |
| STRING_ERG = AXSTRING (AXIS) | Result type: STRING |

Type conversion from STRING:

| BOOL_ERG = ISNUMBER (STRING) | Result type: BOOL |
| REAL_ERG = NUMBER (STRING) | Result type: REAL |
| AXIS_ERG = AXNAME (STRING) | Result type: AXIS |

Concatenation of strings:

| bel._Typ" << bel. Typ" | Result type: STRING |

Conversion to lower/upper case:

| STRING_ERG = TOUPPER (STRING) | Result type: STRING |
| STRING_ERG = TOLOWER (STRING) | Result type: STRING |

Length of the string:

| INT_ERG = STRLEN (STRING) | Result type: INT |

Look for character/string in the string:

| INT_ERG = INDEX (STRING, CHAR) | Result type: INT |
| INT_ERG = RINDEX (STRING, CHAR) | Result type: INT |
| INT_ERG = MINDEX (STRING, STRING) | Result type: INT |
| INT_ERG = MATCH (STRING, STRING) | Result type: INT |

Selection of a substring:

| STRING_ERG = SUBSTR (STRING, INT) | Result type: INT |
| STRING_ERG = SUBSTR (STRING, INT, INT) | Result type: INT |

Selection of a single character:

| CHAR_ERG = STRINGVAR [IDX] | Result type: CHAR |
| CHAR_ERG = STRINGFELD [IDX_FELD, IDX_CHAR] | Result type: CHAR |

1) "bel._Typ" stands for variable types INT, REAL, CHAR, STRING, and BOOL.
Special meaning of the 0 char

The 0 char is interpreted internally end-of-string.
Replacing a character by the 0 character truncates the string.

Example:

```
DEF STRING[20] STRG = "Axis . stopped"
STRG[6] = "X" ;Returns the message "Axis X stopped"
MSG(STRG)
STRG[6] = 0
MSG(STRG) ;Returns the message "Axis"
```

1.10.1 Type conversion

This enables use of variables of different types in a message (MSG).

**Conversion to STRING**

Performed implicitly with use of the operator $<$ for data types INT, REAL, CHAR, and BOOL (see "Concatenation of strings").
An INT value is converted to normal readable format. REAL values convert with up to 10 decimal places.

Variables of type AXIS can be converted to STRING by the AXSTRING function.
FRAME variables cannot be converted.

Example:
```
MSG("Position:"<<$AA_IM[X])
```
Conversion from STRING
The NUMBER function converts from STRING to REAL.
If ISNUMBER returns the value FALSE, CALLING NUMBER with the same parameter will trigger an alarm.
The AXNAME function converts a string to data type AXIS. An alarm is output if the string cannot be assigned to any configured axis identifier.

Syntax

<table>
<thead>
<tr>
<th>Function</th>
<th>Result type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL_ERG = ISNUMBER (STRING)</td>
<td>Result type: BOOL</td>
</tr>
<tr>
<td>REAL_ERG = NUMBER (STRING)</td>
<td>Result type: REAL</td>
</tr>
<tr>
<td>STRING_ERG = AXSTRING (AXIS)</td>
<td>Result type: STRING</td>
</tr>
<tr>
<td>AXIS_ERG = AXNAME (STRING)</td>
<td>Result type: AXIS</td>
</tr>
</tbody>
</table>

Semantics:
ISNUMBER (STRING) returns TRUE, if the string is a valid REAL by the rules of the language. It is thus possible to check whether the string can be converted to a valid number.
NUMBER (STRING) returns the number represented by the string as a REAL.
AXSTRING (AXIS) returns the specified axis identifier as a string.
AXNAME (STRING) converts the string specified to an axis identifier.

Examples

```plaintext
DEF BOOL BOOL_ERG
DEF REAL REAL_ERG
DEF AXIS AXIS_ERG
DEF STRING[32] STRING_ERG

BOOL_ERG = ISNUMBER ("1234.9876Ex-7") ;Now: BOOL_ERG == TRUE
BOOL_ERG = ISNUMBER ("1234XYZ") ;Now: BOOL_ERG == FALSE
REAL_ERG = NUMBER ("1234.9876Ex-7") ;Now: REAL_ERG == 1234.9876Ex-7
STRING_ERG = AXSTRING(X) ;Now: STRING_ERG == "X"
AXIS_ERG = AXNAME("X") ;Now: AXIS_ERG == X
```
1.10.2 Concatenation of strings

This functionality puts a string together out of separate components. The chaining function is implemented via operator: <<. This operator has STRING as the target type for all combinations of basic types CHAR, BOOL, INT, REAL and STRING. Any conversion that may be required is carried out according to existing rules. Types FRAME and AXIS cannot be used with this operator.

**Syntax:**
```
bel._Typ << bel._Typ
```

**Result type:** STRING

**Semantics:**
The strings specified (possibly implicitly converted non-string types) are concatenated.

This operator can also be used as a "unary" operator with a single operand. This can be used for explicit type conversion to STRING (not for FRAME and AXIS).

**Syntax:**
```
<< bel._Typ
```

**Result type:** STRING

**Semantics:**
The specified type is implicitly converted to STRING type.

This can be used to put together a message or a command out of text lists and insert parameters into it (e.g. a module name):
```
MSG(STRG_TAB[LOAD_IDX]<<MODULE_NAME)
```
The intermediate results of string concatenation must not exceed the maximum string length.

### Programming examples

```plaintext
DEF INT IDX = 2
DEF REAL VALUE = 9.654
DEF STRING[20] STRG = "INDEX:2"
IF STRG == "Index:" KE IDX GOTOF NO_MSG
MSG ("Index:" KE IDX KE "\nValue:" KE VALUE) ; Display: "Index: 2\nValue: 9.654"
NO_MSG:
```

### 1.10.3 Conversion to lower/upper case

This functionality permits conversion of all letters of a string to standard capitalization.

#### Syntax:

```plaintext
STRING_ERG = TOUPPER (STRING) Result type: STRING
STRING_ERG = TOLOWER (STRING) Result type: STRING
```

#### Semantics:

All lower case letters are converted to either upper or lower case letters.

#### Example:

Because user inputs can be initiated on the MMC, they can be given standard capitalization (upper or lower case):

```plaintext
DEF STRING [29] STRG ...
IF "LEARN.CNC" == TOUPPER (STRG) GOTOF LOAD_LEARN
```


1.10.4 Length of the string

This functionality sets the length of a string.

Syntax:

\[
\text{INT\_ERG} = \text{STRLEN} (\text{STRING}) \\
\text{Result type: INT}
\]

Semantics:

It returns a number of characters that are not the 0 character, counting from the beginning of the string.

Example:

This can be used to ascertain the end of the string, for example, in conjunction with the single character access described below:

\[
\text{IF} (\text{STRLEN} (\text{MODULE\_NAME}) > 10) \text{ GOTOF ERROR}
\]

1.10.5 Search for character/string in a string

This functionality searches for single characters or a string within a string. The function results specify where the character/string is positioned in the string that has been searched.

\[
\begin{align*}
\text{INT\_ERG} &= \text{INDEX} \ (\text{STRING}, \text{CHAR}) \quad \text{Result type: INT} \\
\text{INT\_ERG} &= \text{RINDEX} \ (\text{STRING}, \text{CHAR}) \quad \text{Result type: INT} \\
\text{INT\_ERG} &= \text{MINDEX} \ (\text{STRING}, \text{STRING}) \quad \text{Result type: INT} \\
\text{INT\_ERG} &= \text{MATCH} \ (\text{STRING}, \text{STRING}) \quad \text{Result type: INT}
\end{align*}
\]

Semantics:

Search functions: They return the position in the string (first parameter) where the search has been successful. If the character/string cannot be found, the value "–1" is returned. In this case, the first character is in position 0.
INDEX searches for the character specified as the second parameter in the string specified as the second parameter (from the beginning).

RINDEX searches for the character specified as the second parameter in the string specified as the second parameter (from the end).

MINDEX same as the INDEX function except that a list of characters is specified (as a string) and the index of the first character found is returned.

MATCH searches for a string in a string.

This can be used to break up a string by certain criteria, for example, at blanks or path separators ("/").

**Programming example**

Example of breaking up an input string into path and module names:

```plaintext
DEF INT PATHIDX, PROGIDX
DEF STRING[26] INPUT
DEF INT LISTIDX

INPUT = "/_N_MPF_DIR/_N_EXECUTE_MPF"

LISTIDX = MINDEX (INPUT, "M,N,O,P") +1

The value returned in LISTIDX is 3 because "N" is the first char from the selection list in parameter INPUT, searching from the beginning.

PATHIDX = INDEX (INPUT, "/") +1

;Therefore: PATHIDX = 1

PROGIDX = RINDEX (INPUT, "/") +1

;Therefore: PROGIDX = 1

;The SUBSTR function introduced in the next section can be used to break up variable INPUT into the components "Path" and "Module":

VARIABLE = SUBSTR (INPUT, PATHIDX, PROGIDX-PATHIDX-1)
returning "/_N_MPF_DIR"

VARIABLE = SUBSTR (INPUT, PROGIDX)
returning "/_N_EXECUTE_MPF"
```
1.10.6 Selection of a substring

This functionality extracts a substring from a string. For this purpose, the index of the first character and the desired string length (if applicable) are specified. If no length information is specified, then the string data refers to the remaining string.

**STRING_ERG = SUBSTR (STRING, INT)**  
Result type: INT

**STRING_ERG = SUBSTR (STRING, INT, INT)**  
Result type: INT

**Semantics:**

In the first case, the substring from the position specified in the first parameter to the end of the string is returned. In the second case, the result string goes up to the maximum length specified in the third parameter. If the initial position is after the end of the string, the empty string ("") will be returned. A negative initial position or length triggers an alarm.

**Example:**

```
def STRING [29] ERG
ERG = SUBSTR ("ACK: 10 to 99", 10, 2) ;Therefore: ERG == "10"
```
1.10.7 Selection of a single character

This functionality selects a single character from a string. This applies both to read access and write access operations.

Syntax:

```
CHAR_ERG = STRINGVAR [IDX]  
Result type: CHAR
CHAR_ERG = STRINGARRAY [IDX_FELD, IDX_CHAR]  
Result type: CHAR
```

Semantics:

The character at the specified position is read/written within the string. If the position parameter is negative or greater than the string, then an alarm is output.

Example messages:

Insertion of an axis identifier into a prepared string.

```
DEF STRING [50] MESSAGE = "Axis n has reached position"
MESSAGE [6] = "X"
MSG (MESSAGE)  \returns message "Axis X has reached position"
```
Single character access is possible only to user-defined variables (LUD, GUD, and PUD data). This type of access is also possible only for "call-by-value" type parameters in subprogram calls.

Examples:

**Single character access to a system, machine data, ...:**

```plaintext
DEF STRING [50] STRG
DEF CHAR ACK
...
STRG = $P_MMCA
ACK = STRG [0] ; Evaluation of acknowledgment component
```

**Single character access in call-by-reference parameter:**

```plaintext
DEF STRING [50] STRG
DEF CHAR CHR1
EXTERN UP_CALL (VAR CHAR1) ; Call-by-reference parameter!
...
CHR = STRG [5]
UP_CALL (CHR1) ; Call-by-reference
STRG [5] = CHR1
```
1.11 CASE instruction

Programming

```
CASE (expression) OF constant1 GOTOF LABEL1 ... DEFAULT GOTOF LABELn
CASE (expression) OF constant1 GOTOB LABEL1 ... DEFAULT GOTOB LABELn
```

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE</td>
<td>Vocabulary word for jump instruction</td>
</tr>
<tr>
<td>GOTOB</td>
<td>Jump instruction with jump destination backward (towards the start of program)</td>
</tr>
<tr>
<td>GOTOF</td>
<td>Jump instruction with jump destination forward (towards the end of program)</td>
</tr>
<tr>
<td>GOTO</td>
<td>Jump instruction with the jump destination first forward and then backward (the direction first to the end of the program and then to the start of the program)</td>
</tr>
<tr>
<td>GOTOC</td>
<td>Suppress alarm 14080 &quot;Jump destination not found&quot;. Jump instruction with the jump destination first forward and then backward (the direction first to the end of the program and then to the start of the program)</td>
</tr>
<tr>
<td>LABEL</td>
<td>Destination (label within the program)</td>
</tr>
<tr>
<td>LABEL:</td>
<td>The name of the jump destination is followed by a colon</td>
</tr>
<tr>
<td>Expression</td>
<td>Arithmetic expression</td>
</tr>
<tr>
<td>Constant</td>
<td>Constant of type INT</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>Program path if none of the previously named constants applies</td>
</tr>
</tbody>
</table>

Function

The CASE statement enables various branches to be executed according to a value of type INT.

Sequence

The program jumps to the point specified by the jump destination, depending on the value of the constant evaluated in the CASE statement.

For more information on the GOTO commands, see Chapter 10, Arithmetic parameters and programm jumps.
In cases where the constant matches none of the predefined values, the DEFAULT instruction can be used to determine the jump destination.

If the DEFAULT instruction is not programmed, the jump destination is the block following the CASE statement.

**Programming example**

**Example 1**

```
CASE(expression) OF 1 GOTOF LABEL1 2 GOTOF LABEL2 ... DEFAULT GOTOF LABELn
```

"1" and "2" are possible constants.

If the value of the expression = 1 (INT constant), jump to block with LABEL1

If the value of the expression = 2 (INT constant), jump to block with LABEL2

... otherwise jump to the block with LABELn

**Example 2**

```
DEF INT VAR1 VAR2 VAR3
CASE(VAR1+VAR2-VAR3) OF 7 GOTOF LABEL1 9 GOTOF LABEL2 DEFAULT GOTOF LABEL3
LABEL1: G0 X1 Y1
LABEL2: G0 X2 Y2
LABEL3: G0 X3 Y3
```
1.12 Control structures

Explanation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF–ELSE–ENDIF</td>
<td>Selection between 2 alternatives</td>
</tr>
<tr>
<td>LOOP–ENDLOOP</td>
<td>Endless loop</td>
</tr>
<tr>
<td>FOR–ENDFOR</td>
<td>Count loop</td>
</tr>
<tr>
<td>WHILE–ENDWHILE</td>
<td>Loop with condition at beginning of loop</td>
</tr>
<tr>
<td>REPEAT–UNTIL</td>
<td>Loop with condition at end of loop</td>
</tr>
</tbody>
</table>

Function

The control processes the NC blocks as standard in the programmed sequence.

In addition to the program branches described in this Chapter, these commands can be used to define additional alternatives and program loops.

These commands enable the user to produce well-structured and easily legible programs.

Sequence

1. IF–ELSE–ENDIF

An IF–ELSE–ENDIF block is used to select one of two alternatives:

**IF** (expression)
NC blocks
**ELSE**
NC blocks
**ENDIF**

If the value of the expression is TRUE, i.e. the condition is fulfilled, then the next program block is executed. If the condition is not fulfilled, then the ELSE program branch is executed. The ELSE branch can be omitted.
2. Endless program loop LOOP

Endless loops are used in endless programs. At the end of the loop, there is always a branch back to the beginning.

```
LOOP
NC blocks
ENDLOOP
```

3. Count loop FOR

The FOR loop is used if it is necessary to repeat an operation by a fixed number of runs. In this case, the count variable is incremented from the start value to the end value. The start value must be lower than the end value. The variable must be of the INT type.

```
FOR Variable = start value TO end value
NC blocks
ENDFOR
```

4. Program loop with condition at start of the loop WHILE

The WHILE program loop is executed for as long as the condition is fulfilled.

```
WHILE expression
NC blocks
ENDWHILE
```
5. Program loop with condition at end of loop

REPEAT

The REPEAT loop is executed once and repeated continuously until the condition is fulfilled.

REPEAT
NC blocks
UNTIL (expression)

Nesting depth

Check structures apply locally within programs. A nesting depth of up to 8 check structures can be set up on each subprogram level.

Runtime response

In interpreter mode (active as standard), it is possible to shorten program processing times more effectively by using program branches than can be obtained with check structures. There is no difference between program branches and check structures in precompiled cycles.
1.12 Control structures

Supplementary conditions

Blocks with check structure elements cannot be suppressed. Labels may not be used in blocks of this type.

Check structures are processed interpretively. When a loop end is detected, a search is made for the loop beginning, allowing for the check structures found in the process. For this reason, the block structure of a program is not checked completely in interpreter mode. It is not generally advisable to use a mixture of check structures and program branches. A check can be made to ensure that check structures are nested correctly when cycles are preprocessed.

Check structures may only be inserted in the statement section of a program. Definitions in the program header may not be executed conditionally or repeatedly.

It is not permissible to superimpose macros on vocabulary words for check structures or on branch destinations. No such check is made when the macro is defined.

Programming example

1. Endless program

```
%_N_LOOP_MPF

LOOP
  IF NOT $P_SEARCH ;No block search
  G01 G90 X0 Z10 F1000
  WHILE $AA_IM[X] <= 100
    G1 G91 X10 F500 ;Drilling pattern
    Z–5 F100
    Z5
```
1.12 Control structures

ENDWHILE
Z10
ELSE ;Block search
MSG("No drilling during block search")
ENDIF
$A_OUT[1]=1 ;Next drilling plate
G4 F2
ENDIF
G4 F2
ENDLOOP
M30

2. Production of a fixed quantity of parts

%_N_WKPCCOUNT_MPF

DEF INT WKPCOUNT
FOR WKPCOUNT = 0 TO 100
    G01 ...
ENDFOR
M30
1.13 Program coordination

Channels

A channel can process its own program independently of other channels. It can control the axes and spindles temporarily assigned to it via the program.

Two or more channels can be set up for the control during startup.

Program coordination

If several channels are involved in the machining of a workpiece it may be necessary to synchronize the programs. Special instructions (commands) are available for program coordination. Each instruction is programmed separately in a block.

Note

Program coordination in the own channel is possible from SW 5.3.
Instructions for program coordination

- **Specification with absolute path**
  
  The absolute path is programmed according to the following rules:
  
  - *Current directory/*_name_MPF
    
    "current directory" stands for the selected workpiece directory or the standard directory /_N_MPF_DIR.
  
  - Selects a particular program for execution in a particular channel:
    
    n: Number of the channel, value per control configuration
  
  - Complete program name
    
    **SW 3 and lower:**
    
    At least one executable block must be programmed between an init command (without synchronization) and an NC start. With subprogram calls "_SPF" must be added to the path.

Example:

```
INIT(2,"/_N_WKS_DIR/_ABRICHT_MPF")
G01 F0.1
START
```

```
INIT(2,"/_N_WCS_DIR/_N_UNDER_1_SPF")
```

- **Relative path specification**

  The same rules apply to relative path definition as for program calls.

Example:

```
INIT(2,"DRESS")
```

```
INIT(3,"UNDER_1_SPF")
```

```
START (n,n)
```

```
WAITM (Marker No.,n,n,...)
```

Sets the marker "Marker No." in the same channel. Terminate previous block with exact stop. Waits for the markers with the same "Marker no." in the specified channels "n" (current channel does not have to be specified). Marker is deleted after synchronization. 10 markers can be set per channel simultaneously.
### 1.13 Program coordination

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAITMC(Marker No., n, n, ...)</td>
<td>Sets the marker &quot;Marker No.&quot; in the same channel. An exact stop is initiated only if the other channels have not yet reached the marker. Waits for the marker with the same &quot;Marker No.&quot; in the specified channels &quot;n&quot; (current channel does not have to be specified). As soon as marker &quot;Marker No.&quot; in the specified channels is reached, continue without terminating exact stop.</td>
</tr>
<tr>
<td>WAITE(n,n)</td>
<td>Waits for the end of program of the specified channels (current channel not specified)</td>
</tr>
<tr>
<td>SETM(Marker No., Marker No., ...)</td>
<td>Sets the markers &quot;Marker No.&quot; in the same channel without affecting current processing. SETM() remains valid after RESET and NC START. SETM() can also be programmed independently of a synchronized action.</td>
</tr>
<tr>
<td>CLEARM(Marker No., Marker No., ...)</td>
<td>Deletes the markers &quot;Marker No.&quot; in the same channel without affecting current processing. All markers can be deleted with CLEARM(). CLEARM(0) deletes the marker &quot;0&quot;. CLEARM() remains valid after RESET and NC START. CLEARM() can also be programmed independently of a synchronized action.</td>
</tr>
</tbody>
</table>

**Note**

All the above commands must be programmed in separate blocks. The number of markers depends on the CPU used.

**Channel names**

Channel names must be converted to numbers via variables (see Chapter 10 "Variables and Arithmetic Parameters").

*Protect the number assignments so that they are not changed unintentionally.*
Example:
Channel called "MACHINE" is to contain channel number 1,
Channel called "LOADER" is to contain channel number 2,
DEF INT MACHINE=1, LOADER=2
The variables are given the same names as the channels.
The instruction START is therefore:
START(MACHINE)

Example of program coordination

Channel 1:

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>INIT(2,&quot;MPF200&quot;)</td>
<td>Program execution in channel 2</td>
</tr>
<tr>
<td>N11</td>
<td>START (2)</td>
<td>Wait for WAIT mark 1 in channel 1 and in channel 2 and execution continued in channel 1</td>
</tr>
<tr>
<td>N80</td>
<td>WAITEM(1,1,2)</td>
<td>Wait for WAIT mark 2 in channel 1 and in channel 2 and execution continued in channel 1</td>
</tr>
<tr>
<td>N180</td>
<td>WAITEM(2,1,2)</td>
<td>Wait for end of program in channel 2</td>
</tr>
<tr>
<td>N201</td>
<td>M30</td>
<td>End of program channel 1, end all</td>
</tr>
</tbody>
</table>

Channel 2:

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N70</td>
<td>WAITEM(1,1,2)</td>
<td>Program execution in channel 2</td>
</tr>
<tr>
<td>N270</td>
<td>WAITEM(2,1,2)</td>
<td>Wait for WAIT mark 1 in channel 1 and in channel 2 and execution continued in channel 2</td>
</tr>
<tr>
<td>N400</td>
<td>M30</td>
<td>End of program in channel 2</td>
</tr>
</tbody>
</table>
Example of program from workpiece

N10 INIT(2,"/_N_WKS_DIR/_N_SHAFT1_WPD/_N_CUT1_MPF")

Example of Init command with relative path definition

; Program /_N_MPF_DIR/_N_MAIN_MPF is selected in channel 1
N10 INIT(2,"MYPROG") ; select program /_N_MPF_DIR/_N_MYPROG_MPF in channel 2.

Additional notes

Variables which all channels can access (NCK-specific global variables) can be used for data exchange between programs. Otherwise separate programs must be written for each channel.

SW 3 and lower:
WAITE must not be scanned immediately after the START command or else a program end will be detected before the program is started.
Remedy: Programming a dwell time.

Example:
N30 START(2)
N31 G4 F0.01
N40 WAITE(2)
1.14 Interrupt routine

Programming

```
SETINT(3) PRIO=1 NAME
SETINT(3) PRIO=1 LIFTFAST
SETINT(3) PRIO=1 NAME LIFTFAST
G... X... Y... ALF=...
DISABLE(3)
ENABLE(3)
CLRINT(3)
```

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETINT(n)</td>
<td>Start interrupt routine if input n is enabled, n (1...8) stands for the number of the input</td>
</tr>
<tr>
<td>PRIO=1</td>
<td>Define priority 1 to 128 (1 has top priority)</td>
</tr>
<tr>
<td>LIFTFAST</td>
<td>Fast lift from contour</td>
</tr>
<tr>
<td>NAME</td>
<td>Name of the subprogram to be executed</td>
</tr>
<tr>
<td>ALF=...</td>
<td>Programmable traverse direction (in motion block)</td>
</tr>
<tr>
<td>DISABLE(n)</td>
<td>Deactivate interrupt routine number n</td>
</tr>
<tr>
<td>ENABLE(n)</td>
<td>Reactivate interrupt routine number n</td>
</tr>
<tr>
<td>CLRINT(n)</td>
<td>Clear interrupt assignments of interrupt routine number n</td>
</tr>
</tbody>
</table>

Function

Example: The tool breaks during machining. This triggers a signal that stops the current machining process and simultaneously starts a subprogram – this subprogram is called an interrupt routine. The interrupt routine contains all the instructions which are to be executed in this case. When the interrupt routine has finished being executed and the machine is ready to continue operation, the control jumps back to the main program and continues machining at the point of interruption – depending on the REPOS command.

For further information on REPOS, see Chapter 9, Path Traversing Behavior, Repositioning.
1.14 Interrupt routine

Sequence

Create interrupt routine
The interrupt routine is identified as a subprogram in the definition.

Example:
PROC LIFT_Z
N10...
N50 M17

Program name LIFT_Z, followed by the NC blocks, finally end-of-program M17 and return to main program.

Note:
SETINT instructions can be programmed within the interrupt routine and used to activate additional interrupt routines. They are triggered via the input.

You will find more information on how to create subprograms in Chapter 2.

Save interrupt position, SAVE
The interrupt routine can be identified with SAVE in the definition.

Example:
PROC LIFT_Z SAVE
N10...
N50 M17

At the end of the interrupt routine the modal G functions are set to the value they had at the start of the interrupt routine by means of the SAVE attribute. The programmable zero offset and the basic offset are reestablished in addition to the settable zero offset (modal G function group 8). If the G function group 15 (feed type) is changed, e.g. from G94 to G95, the appropriate F value is also reestablished.

Machining can thus be resumed later at the point of interruption.
Assign and start interrupt routine

The control has signals (inputs 1...8) to interrupt the program run and start the corresponding interrupt routine.

The assignment of input to program is made in the main program.

Example:
N10 SETINT(3) PRIO=1 LIFT_Z

When input 3 is enabled, routine LIFT_Z is started immediately.

Start several interrupt routines, define the priority, PRIO=

If several SETINT instructions are programmed in your NC program and several signals can therefore occur at the same time, you must assign the priority of the interrupt routines to determine the order in which they are executed: Priority levels PRIO 1 to 128 are available, 1 has top priority.

Example:
N10 SETINT(3) PRIO=1 LIFT_Z
N20 SETINT(2) PRIO=2 LIFT_X

The routines are executed successively in the order of their priority if the inputs are enabled at the same time. First SETINT(3), then SETINT(2).

If new signals are received while interrupt routines are being executed, the current interrupt routines are interrupted by routines with higher priority.

Deactivate/reactivate interrupt routine

DISABLE, ENABLE

You can deactivate interrupt routines in the NC program with DISABLE(n) and reactive them with ENABLE(n) (n stands for the input number).
1.14 Interrupt routine

The input/routine assignment is retained with DISABLE and reactivated with ENABLE.

Reassign interrupt routines
If a new routine is assigned to an assigned input, the old assignment is automatically canceled.

Example:
N20 SETINT(3) PRIO=2 LIFT_Z
…
…
N120 SETINT(3) PRIO=1 LIFT_X

Clear assignment, CLRINT
Assignments can be cleared with CLRINT(n).

Example:
N20 SETINT(3) PRIO=2 LIFT_Z
N50 CLRINT(3)

The assignment between input 3 and the routine LIFT_Z is cleared.

Rapid lift from contour
When the input is switched, LIFTFAST retracts the tool rapidly from the workpiece contour.

If the SETINT instruction includes an interrupt routine as well as LIFTFAST, the liftfast is executed before the interrupt routine.

Example:
N10 SETINT(2) PRIO=1 LIFTFAST
or
N30 SETINT(2) PRIO=1 LIFT_Z LIFTFAST

In both cases, the liftfast is executed when input 2 with top priority is enabled.
- With N10, execution is stopped with alarm 16010 (as no asynchronized subprogram, ASUB, was specified).
• The asynchronized subprogram "LIFT-Z" is executed with N30.

When determining the lift direction, a check is performed to see whether a frame with mirror is active. If one is active, right and left are inverted for the lift direction with regard to the tangent direction. The direction components in tool direction are not mirrored. This behavior is activated via MD $MC_LIFTFAST_WITH_MIRROR=TRUE

Sequence of motions with rapid lift
The distance through which the geometry axes are retracted from the contour on liftfast can be defined in machine data.

Programmable traversing direction, ALF=...
You enter the direction in which the tool is to travel on liftfast in the NC program.

The possible traversing directions are stored in special code numbers on the control and can be called up using these numbers.
Example:
N10 SETINT(2) PRIO=1 LIFT_Z LIFTFAST ALF=7

The tool moves – with G41 activated (direction of machining to the left of the contour) – away from the contour perpendicularly as seen from above.
Reference plane for describing the traversing directions
At the point of application of the tool to the programmed contour, the tool is clamped at a plane which is used as a reference for specifying the liftoff movement with the corresponding code number.

The reference plane is derived from the longitudinal tool axis (infeed direction) and a vector positioned perpendicular to this axis and perpendicular to the tangent at the point of application of the tool.

Code number with traversing directions, overview
The code numbers and the traversing directions in relation to the reference plane are shown in the diagram on the right.

ALF=0 deactivates the liffast function.

Please note:
The following codes should not be used when tool radius compensation is active:
Codes 2, 3, 4 with G41
Codes 6, 7, 8 with G42.

In these cases, the tool would approach the contour and collide with the workpiece.
Retraction movement in SW 4.3 and higher
The direction of the retraction movement is programmed by means of the G code LFTXT or LFWP with the variable ALF.

- **LFTXT**
  The plane of the retraction movement is determined from the path tangent and the tool direction. This G code (default setting) is presently used for programming the behavior for fast lift.

- **LFWP**
  The plane for the retraction movement is the active working plane which is selected by means of G codes G17, G18 or G19. The direction of the retraction movement is not dependent on the path tangent. Thus it is possible to program an axis-parallel fast lift.

In the retraction movement plane, ALF is used to program the direction in discrete steps of 45 degrees as was the case formerly. With LFTXT retraction in tool direction is defined for ALF=1.

With LFWP the direction in the working plane is according to the following:

- **G17**: X/Y plane  ALF=1 retraction in X direction  ALF=3 retraction in Y direction
- **G18**: Z/X plane  ALF=1 retraction in Z direction  ALF=3 retraction in X direction
- **G19**: Y/Z plane  ALF=1 retraction in Y direction  ALF=3 retraction in Z direction
Programming example

In this example, a broken tool is to be replaced automatically by an alternate tool. Machining is continued with the new tool. Machining is then continued with the new tool.

Main program

```
N10 SETINT(1) PRIO=1 W_CHANGE ->
-> LIFTFAST

When input 1 is enabled, the tool is automatically retracted from the contour with lifffast (code no. 7 for tool radius compensation G41). Interrupt routine W_CHANGE is subsequently executed.

N20 G0 Z100 G17 T1 ALF=7 D1
N30 G0 X-5 Y-22 Z2 M3 S300
N40 Z-7
N50 G41 G1 X16 Y16 F200
N60 Y35
N70 X53 Y65
N90 X71.5 Y16
N100 X16
N110 G40 G0 Z100 M30
```

Subprogram

```
PROC W_CHANGE SAVE

Subprogram with storage of current operating state

N10 G0 Z100 M5

Tool changing position, spindle stop

N20 T11 M6 D1 G41

Change tool

N30 REPOS RMB M3

Repositioning and return to main program
```

-> programmed in a single block.

If you do not program any of the REPOS commands in the subprogram, the axis is moved to the end of the block that follows the interrupted block.
1.15 Axis transfer, spindle transfer

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELEASE(axis name, axis name, ...)</td>
<td>Enable the axis</td>
</tr>
<tr>
<td>GET(axis name, axis name, ...)</td>
<td>Accept the axis</td>
</tr>
<tr>
<td>GETD (axis name, axis name, ...)</td>
<td>Direct acceptance of axis</td>
</tr>
</tbody>
</table>

Axis name

Axis assignment in system: AX1, AX2, ... or specify machine axis name

RELEASE(S1)
Enable spindles S1, S2, ...

GET(S2)
Accept spindles S1, S2, ...

GETD(S3)
Direct acceptance of spindles S1, S2, ...

Function

One or more axes or spindles can only ever be used in one channel. If an axis has to alternate between two different channels (e.g. pallet changer) it must first be enabled in the current channel and then transferred to the other channel: The axis is transferred from channel to channel.

For more information on the functionality of an axis or spindle replacement, see

/FB/, K5 Mode groups, channels, axis transfer

Sequence

Preconditions for axis transfer

- The axis must be defined by machine data in all the channels that want to use the axis.
- The channel to which the axis is assigned after power ON is defined in the axis-specific machine data.
Release axis: RELEASE
When enabling the axis please note:
1. The axis must not involved in a transformation.
2. All the axes involved in an axis link (tangential control) must be enabled.
3. A concurrent positioning axis cannot be replaced in this situation.
4. All the following axes of a gantry master axis are transferred with the master.
5. With coupled axes (coupled motion, leading value coupling, electronic gear) only the leading axis of the group can be enabled.

Transfer axis: GET
The actual axis transfer is performed with this command. The channel for which the command is programmed takes full responsibility for the axis.

Effects of GET:
Axis transfer with synchronization:
An axis always has to be synchronized if it has been assigned to another channel or the PLC in the meantime and has not been resynchronized with "WAITP", G74 or delete distance-to-go before GET.
- A preprocess stop follows (as for STOPRE)
- Execution is interrupted until the transfer has been completed.
**Axis transfer without synchronization:**

If the axis does not have to be synchronized no preprocess stop is generated by GET.

Example:

```
N01 G0 X0
N02 RELEASE(AX5)
N03 G64 X10
N04 X20
N05 GET(AX5)
```

If synchronization not necessary, this is not an executable block.

```
N06 G01 F5000
N07 X20
```

Not an executable block.

```
N08 X30
N09 ...
```

Not an executable block because X position as for N04.

First executable block after N05.

**Automatic "GET"**

If an axis is in principle available in a channel but is not currently defined as a "channel axis", GET is executed automatically. If the axis/axes is/are already synchronized no preprocess stop is generated.

**An axis accepted with GET remains assigned to this axis even after a key or program reset. When a program is started the transferred axes or spindles must be reassigned in the program if the axis is required in its original channel.**

It is assigned to the channel defined in the machine data on power ON.

**Direct axis transfer: GETD**

An axis is taken directly from another channel with GETD (GET Directly). This means that no matching RELEASE has to be programmed in another channel for this GETD. It also means that other channel communication has to be established (e.g. wait markers).
Programming example

Of the 6 axes, the following are used for machining in channel 1: 1st, 2nd, 3rd and 4th. The 5th and 6th axes in channel 2 are used for the workpiece change.

Axis 2 is to be transferred between the 2 channels and then assigned to channel 1 after power ON.

Program "MAIN" in channel 1

```plaintext
%_N_MAIN_MPF
INIT (2,"TRANSFER2")
N... START (2)
N... GET (AX2)
...%
N... RELEASE (AX2)
N... WAITM (1,1,2)
N...
N... M30
```

<table>
<thead>
<tr>
<th>%_N_MAIN_MPF</th>
<th>Select program TRANSFER2 in channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT (2,&quot;TRANSFER2&quot;)</td>
<td></td>
</tr>
<tr>
<td>N... START (2)</td>
<td>Start program in channel 2</td>
</tr>
<tr>
<td>N... GET (AX2)</td>
<td>Accept axis AX2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N... RELEASE (AX2)</td>
<td>Enable axis AX2</td>
</tr>
<tr>
<td>N... WAITM (1,1,2)</td>
<td>Wait for wait marker in channel 1 and 2 for synchronizing both channels</td>
</tr>
<tr>
<td>N... M30</td>
<td>Rest of program after axis transfer</td>
</tr>
</tbody>
</table>

Program "Replace2" in channel 2

```plaintext
%_N_TRANSFER2_MPF
N... RELEASE (AX2)
N160 WAITM (1,1,2)
N150 GET (AX2)
N...
N...M30
```

<table>
<thead>
<tr>
<th>%_N_TRANSFER2_MPF</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N... RELEASE (AX2)</td>
<td></td>
</tr>
<tr>
<td>N160 WAITM (1,1,2)</td>
<td>Wait for wait marker in channel 1 and 2 for synchronizing both channels</td>
</tr>
<tr>
<td>N150 GET (AX2)</td>
<td>Accept axis AX2</td>
</tr>
<tr>
<td>N... M30</td>
<td>Rest of program after axis transfer</td>
</tr>
</tbody>
</table>
Set up variable axis transfer response
The release time of the axes can be set up using MD 10722: AXCHANGE_MASK as follows:

- Automatic axis transfer between two channels then also takes place when the axis has been brought to a neutral state by WAITP (response as before)
- From SW 5.3, it will only be possible to transfer all the axes fetched to the axis container by GET or GETD after an axis container rotation.
- From SW 6.4, when an intermediate block is inserted in the main run, a check will be made to determine whether or not reorganisation is required. Reorganisation is only necessary if the axis states of this block do not match the current axis states.

Programming example

Activating an axis transfer without a preprocessing stop

N010 M4 S100
N011 G4 F2
N020 M5
N021 SPOS=0
N022 POS[B]=1
N023 WAITP[B]  // Axis B becomes the neutral axis
N030 X1 F10
N031 X100 F500
N032 X200
N040 M3 S500
N041 G4 F2
N050 M5
N099 M30

Traverses the spindle (axis B) immediately after block N023 as the PLC axis e.g. 180 degrees and back 1 degree and back to the neutral axis. So block N040 triggers neither a preprocessing stop nor a reorganization.
1.16 NEWCONF: Setting machine data active (SW 4.3 and higher)

**Function**

All machine data of the effectiveness level "NEW_CONFIG" are set active by means of the NEWCONF language command. The function corresponds to activating the soft key "Set MD active". When the NEWCONF function is executed there is an implicit preprocessing stop, that is, the path movement is interrupted.

**Explanation**

NEWCONF

All machine data of the "NEW_CONFIG" effectiveness level are set active

**Programming example**

Milling operation: Machining drilling position with different technologies

```
N10 $MA_CONTOUR_TOL[AX]=1.0 ; Change machine data
N20 NEWCONF ; Set machine data active
```
1.17 WRITE: Write file (SW 4.3 and higher)

Programm

WRITE(var int error, char[160] filename, char[200] string)

The WRITE command appends a block to the end of the specified file.

Explanation of the parameters

<table>
<thead>
<tr>
<th>error</th>
<th>Error variable for return</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Path not allowed</td>
</tr>
<tr>
<td>2</td>
<td>Path not found</td>
</tr>
<tr>
<td>3</td>
<td>File not found</td>
</tr>
<tr>
<td>4</td>
<td>Incorrect file type</td>
</tr>
<tr>
<td>10</td>
<td>File is full</td>
</tr>
<tr>
<td>11</td>
<td>File is being used</td>
</tr>
<tr>
<td>12</td>
<td>No free resources</td>
</tr>
<tr>
<td>13</td>
<td>No access rights</td>
</tr>
<tr>
<td>20</td>
<td>Other error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>filename</th>
<th>Name of file in which the string is to be written.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The file name can be specified with path and file identifier. Path names must be absolute, that is, starting with &quot;/&quot;. If the file name does not contain a domain identifier (<em>N</em>), it is added accordingly. If there is not identifier (_MPF or _SPF), the file name is automatically completed with _MPF. If there is no path specified, the file is saved in the current directory (= directory of selected program). The file name length can be up to 32 bytes, the path length up to 128 bytes.</td>
</tr>
<tr>
<td></td>
<td>Example: PROTFILE</td>
</tr>
<tr>
<td></td>
<td>_N_PROTFILE</td>
</tr>
<tr>
<td></td>
<td>_N_PROTFILE_MPF</td>
</tr>
<tr>
<td></td>
<td>/_N_MPF_DIR/_N_PROTFILE_MPF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>Text to be written. Internally LF is then added; this means that the text is lengthened by one character.</th>
</tr>
</thead>
</table>
Function

Using the WRITE command, data (e.g. measurement results for measuring cycles) can be appended to the end of the specified file. The maximum length in KB of the log files is set via MD 11420 LEN_PROTOCOL_FILE. This length is applicable for all files created using the WRITE command.

Once the file reaches the specified length, an error message is output and the string is not saved. If there is sufficient free memory, a new file can be created. The created files can be
• read, edited and deleted by all users,
• written in the parts program that is currently being executed. The blocks are inserted at the end of the file, after M30.

Programming example

```
N10 DEF INT ERROR ;
N20 WRITE(ERROR,"TEST1","LOG FROM 7.2.97") ; Write text from LOG FROM 7.2.97 in the file TEST1
N30 IF ERROR ;
N40 MSG ("Error with WRITE command:" <<ERROR) ;
N50 M0 ;
N60 ENDIF ;
...
WRITE(ERROR, "/_N_WCS_DIR/_N_PROT_WPD/_N_PROT_MPF", "LOG FROM 7.2.97") ; Absolute path
```

Additional notes

• If no such file exists in the NC, it is newly created and can be written to by means of the WRITE command.
1.18 DELETE: Delete file (SW 4.3 and higher)

Programming

DELETE(var int error, char[160] filename)

The DELETE command deletes the specified file.

Explanation of the parameters

<table>
<thead>
<tr>
<th>error</th>
<th>Error variable for return</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Path not allowed</td>
</tr>
<tr>
<td>2</td>
<td>Path not found</td>
</tr>
<tr>
<td>3</td>
<td>File not found</td>
</tr>
<tr>
<td>4</td>
<td>Incorrect file type</td>
</tr>
<tr>
<td>11</td>
<td>File is being used</td>
</tr>
<tr>
<td>12</td>
<td>No free resources</td>
</tr>
<tr>
<td>20</td>
<td>Other error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>filename</th>
<th>Name of the file to be deleted</th>
</tr>
</thead>
</table>

The file name can be specified with path and file identifier. Path names must be absolute, that is, starting with "/". If the file name does not contain a domain identifier (_N_), it is added accordingly. The file identifier ("-" plus 3 characters), e.g. _SPF) is optional. If there is no identifier, the file name is automatically added _MPF. If there is no path specified, the file is saved in the current directory (= directory of selected program). The file name length can be up to 32 bytes, the path length up to 128 bytes.
1.19 READ: Read lines in file (SW 5.2 and higher)

Function

All files can be deleted by means of the DELETE command, irrespective of whether they were created using the WRITE command or not. Files that were created using a higher access authorization can also be deleted with DELETE.

Programming example

```plaintext
N10 DEF INT ERROR;
N15 STOPRE; preprocessing stop
N20 DELETE (ERROR,
"/_N_SPF_DIR/_N_TEST1_SPF") ; deletes file TEST1 in the
subroutine branch
N30 IF ERROR ;
N40 MSG ("Error with DELETE command:"
<<ERROR)
N50 M0 ;
N60 ENDIF ;
...```

1.19 READ: Read lines in file (SW 5.2 and higher)

Programming

READ(var int error, string[160] file, int line, int number, var string[255] result[])

The READ command reads one or several lines in the file specified and stores the information read in an array of type STRING. In this array, each read line occupies an array element.
11.02 Flexible NC Programming

1.19 READ: Read lines in file (SW 5.2 and higher)

Explanation of the parameters

<table>
<thead>
<tr>
<th>error</th>
<th>Error variable for return (call-by-reference parameter, type INT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
<tr>
<td>1</td>
<td>Path not allowed</td>
</tr>
<tr>
<td>2</td>
<td>Path not found</td>
</tr>
<tr>
<td>3</td>
<td>File not found</td>
</tr>
<tr>
<td>4</td>
<td>Incorrect file type</td>
</tr>
<tr>
<td>13</td>
<td>Insufficient access rights</td>
</tr>
<tr>
<td>21</td>
<td>Line not available (parameter &quot;line&quot; or &quot;number&quot; larger than number of lines in file)</td>
</tr>
<tr>
<td>22</td>
<td>Array length of &quot;result&quot; variable too small</td>
</tr>
<tr>
<td>23</td>
<td>Line range too large (parameter &quot;number&quot; has been selected so large, that reading goes beyond the end of the file)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>file</th>
<th>Name/path of the file to be read (call-by-value parameter of type STRING with a max. length of 160 bytes). The file must be stored in the user memory of the NCK (passive file system). The file name can be preceded by the domain identifier <em>N</em>. If the domain identifier is missing, it is added correspondingly. The file identifier (&quot;_&quot; plus three characters, e.g. _SPF) is optional. If there is no identifier, the file name is automatically added _MPF. If there is no path specified in &quot;file&quot;, the file is searched for in the current directory (=directory of selected program). If a path is specified in &quot;file&quot;, it must start with a slash &quot;/&quot; (absolute path indication).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>line</th>
<th>Position indication of the line range to be read (call-by-value parameter of type INT).</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The number of lines before the end of the file which is specified by the parameter &quot;number&quot; is read.</td>
</tr>
<tr>
<td>1 to n</td>
<td>Number of the first line to be read.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>number</th>
<th>Number of lines to be read (call-by-value parameter of type INT).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>result</th>
<th>Array of type STRING, where the read text is stored (call-by-reference parameter with a length of 255).</th>
</tr>
</thead>
</table>
1.19 READ: Read lines in file (SW 5.2 and higher)

Function

One or several lines can be read from a file with the READ command. The lines read are stored in one array element of an array. The information is available as STRING.

Additional notes

- Binary files cannot be read in. The error message error=4:Wrong type of file is output. The following types of file are not readable: _BIN, _EXE, _OBJ, _LIB, _BOT, _TRC, _ACC, _CYC, _NCK.
- The currently set protection level must be equal to or greater than the READ right of the file. If this is not the case, access is denied with error=13.
- If the number of lines specified in the parameter "number" is smaller than the array length of "result", the other array elements are not altered.
- Termination of a line by means of the control characters "LF" (Line Feed) or "CR LF" (Carriage Return Line Feed) is not stored in the target variable "result". Read line are cut off, if the line is longer than the string length of the target variable "result". An error message is not output.

Programming example

```
N10 DEF INT ERROR ; error variable
...
N30 READ(ERROR, "TESTFILE", 1, 5, RESULT) ; file name without domain and file identifier
...
N30 READ (ERROR, "TESTFILE_MPF", 1, 5, RESULT) ; file name without domain and with file identifier
...
N30 READ (ERROR,"_N_TESTFILE_MPF",1,5,RESULT) ; file name with domain and file identifier
```
...  
N30 READ(ERROR,"/_N_CST_DIR/N_TESTFILE```_MPF", 1, 5 RESULT)  ; file name with domain and file identifier and path specification  
^...
N40 IF ERROR <>0  ; error evaluation
N50 MSG("ERROR"<<ERROR<<" WITH READ COMMAND")
N60 M0
N70 ENDIF
...

1.20 ISFILE: File available in user memory NCK (SW 5.2 and higher)

Programming

result=isfile(string[160]file)

With the ISFILE command you check whether a file exists in the user memory of the NCK (passive file system). As a result either TRUE (file exists) or False (file does not exist) is returned.

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>Name/path of the file to be read (call-by-value parameter of type STRING with a max. length of 160 bytes). The file must be stored in the user memory of the NCK (passive file system). The file name can be preceded by the domain identifier <em>N</em>. If the domain identifier is missing, it is added correspondingly. The file identifier (&quot;_&quot; plus three characters, e.g. _SPF) is optional. If there is no identifier, the file name is automatically added _MPF. If there is no path specified in &quot;file&quot;, the file is searched for in the current directory (=directory of selected program). If a path is specified in &quot;file&quot;, it must start with a slash &quot;/&quot; (absolute path indication).</td>
</tr>
<tr>
<td>result</td>
<td>Variable for storage of the result of type BOOL (TRUE or FALSE)</td>
</tr>
</tbody>
</table>
1.21 CHECKSUM: Creation of a checksum over an array

Programming example

```
N10 DEF BOOL RESULT
N20 RESULT=ISFILE("TESTFILE")
N30 IF(RESULT==FALSE)
N40 MSG("FILE DOES NOT EXIST")
N50 M0
N60 ENDIF
...
```
or:
```
N30 IF(NOT ISFILE("TESTFILE"))
N40 MSG("FILE DOES NOT EXIST")
N50 M0
N60 ENDIF
...
```

1.21 CHECKSUM: Creation of a checksum over an array
(SW 5.2 and higher)

Programming

```
error=CHECKSUM(var string[16] chksum,string[32]array, int first, int last)
```

The CHECKSUM function forms the checksum over an array.

Explanation of the parameters

<table>
<thead>
<tr>
<th>error</th>
<th>Error variable for return</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Symbol not found</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>No array</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Index 1 too large</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Index 2 too large</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Invalid type of file</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Checksum overflow</td>
<td></td>
</tr>
</tbody>
</table>

`chksum` Checksum over the array as a string (call-by-reference parameter of type String, with a defined length of 16).

The checksum is indicated as a character string of 16 hexadecimal numbers. However, no format characters are indicated.

Example: in MY_CHECKSUM
1.21 CHECKSUM: Creation of a checksum over an array

**array**
Number of the array over which the checksum is to be formed. (Call-by-value parameter of type String with a max. length of 32).
Permissible arrays: 1 or 2-dimensional arrays of types BOOL, CHAR, INT, REAL, STRING
Arrays of machine data are not permissible.

**first**
Column number of start column (optional)

**last**
Column number of end column (optional)

**Function**
With CHECKSUM you form a checksum over an array.
Stock removal application:
Check to see whether the initial contour has changed.

**Additional notes**
The parameters first and last are optional. If no column indices are indicated, the checksum is formed over the whole array.

The result of the checksum is always definite. If an array element is changed, the result string will also be changed.

**Programming example**

```
N10 DEF INT ERROR
N20 DEF STRING[16] MY_CHECKSUM
N30 DEF INT MY_VAR[4,4]
N40 MY_VAR=...
N50 ERROR=CHECKSUM
   (CHECKSUM;"MY_VAR", 0, 2)
   ...
returns in MY_CHECKSUM the value
"A6FC3404E534047C"
```
1.21 CHECKSUM: Creation of a checksum over an array
Subprograms, Macros

2.1 Using subprograms ................................................................. 2-102
2.2 Subprogram with SAVE mechanism .......................................... 2-104
2.3 Subprograms with parameter transfer ........................................ 2-105
2.4 Calling subprograms: L or EXTERN ..................................... 2-109
2.5 Parameterizable subprogram return (SW 6.4 and higher) ............. 2-113
2.6 Subprogram with program repetition: P .................................... 2-117
2.7 Modal subprogram: MCALL ................................................... 2-118
2.8 Calling the subprogram indirectly: CALL ................................. 2-119
2.9 Repeating program sections with indirect programming (SW 6.4 and higher) ....................... 2-120
2.10 Calling up a program in ISO language indirectly: ISOCALL .......... 2-121
2.11 Calling subprogram with path specification and param. PCALL .... 2-122
2.12 Extending a search path for subprogram calls with CALLPATH 
   (SW 6.4 and higher) ............................................................... 2-123
2.13 Suppress current block display: DISPLOF ............................... 2-125
2.14 Single block suppression: SBLOF, SBLON (SW 4.3 and higher) ... 2-126
2.15 Executing external subprogram: EXTCALL (SW 4.2 and higher) ... 2-132
2.16 Subprogram call with M/T function ....................................... 2-136
2.17 Cycles: Setting parameters for user cycles ............................... 2-138
2.18 Macros, DEFINE...AS .......................................................... 2-142
2.1 Using subprograms

What is a subprogram?

In principle, a subprogram has the same structure as a parts program. It consists of NC blocks with traverse commands and switching commands.

In principle, there is no difference between a main program and a subprogram. The subprogram contains either machining cycles or machining sections that must run more than once.

Use of subprograms

Machining sequences that recur are only programmed once in a subprogram. For example, certain contour shapes that occur more than once or machining cycles.

This subprogram can be called and executed in any main program.

Structure of the subprogram

The structure of a subprogram is identical to that of the main program.

In a subprogram it is also possible to program a program header with parameter definitions.
11.02 Subprograms, Macros

2.1 Using subprograms

Nesting depth

Nesting of subprograms
A subprogram can itself contain subprogram calls. The subprograms called can contain further subprogram calls etc. The maximum number of subprogram levels or the nesting depth is 12.

This means:
A main program can contain 11 nested subprogram calls.

Restrictions
It also possible to call subprograms in interrupt routines. For work with subprograms you must keep four levels free or only nest seven subprogram calls.

For SIEMENS machining and measuring cycles you require three levels. If you call a cycle from a subprogram you must do this no deeper than level 5 (if four levels are reserved for interrupt routines).
2.2 Subprogram with SAVE mechanism

Function

For this, specify the additional command SAVE with the definition statement with PROC.

When the subprograms have been executed, the modal G functions are set to the value they had at subprogram start due to the SAVE attribute. If G function group 8 (settable zero offset), G function group 52 (frame rotation of a turnable workpiece), or G function group 53 (frame rotation in tool direction) is changed while doing so, the corresponding frames are restored.

- The active basic frame is not changed when the subprogram returns.
- The programmable zero offset is restored

From SW 6.1 you can change the response of the settable zero offset and the basic frame via machine data MD 10617: FRAME_SAVE_MASK.

You will find more information in
/FB/ K1, General Machine Data

Example:

Subprogram definition
PROC CONTOUR (REAL VALUE1) SAVE
N10 G91 ... 
N100 M17

Main program
%123
N10 G0 X... Y... G90
N20...
N50 CONTOUR (12.4)
N60 X... Y...

In the CONTOUR subprogram G91 incremental dimension applies. After returning to the main program, absolute dimension applies again because the modal functions of the main program were stored with SAVE.
2.3 Subprograms with parameter transfer

Program start, PROC
A subprogram that is to take over parameters from the calling program when the program runs is designated with the vocabulary word PROC.

Subprogram end M17, RET
The command M17 designates the end of subprogram and is also an instruction to return to the calling main program.
As an alternative to M17: The vocabulary word RET stands for end of subprogram without interruption of continuous path mode and without function output to the PLC.

Interruption of continuous-path mode
To prevent continuous-path mode from being interrupted:
Make sure the subprogram does not have the SAVE attribute. For more information about the SAVE mechanism, see Section 2.2.

RET must be programmed in a separate NC block.

Example:
PROC CONTOUR
N10...
...
N100 M17

Parameter transfer between main program and subprogram
If you are working with parameters in the main program, you can use the values calculated or assigned in the subprogram as well.
For this purpose the values of the current parameters of the main program are passed to the formal parameters of the subprogram when the subprogram is called and then processed in subprogram execution.
Example:
N10 DEF REAL LENGTH, WIDTH
N20 LENGTH = 12  WIDTH = 10
N30 BORDER (LENGTH, WIDTH)

The values assigned in N20 in the main program are passed in N30 when the subprogram is called. Parameters are passed in the sequence stated. The parameter names do not have to be identical in the main programs and subprogram.

Two ways of parameter transfer

Values are only passed (call-by-value)
If the parameters passed are changed as the subprogram runs this does not have any effect on the main program. The parameters remain unchanged in it (see Fig.)

Parameter transfer with data exchange (call-by-reference)
Any change to the parameters in the subprogram also causes the parameter to change in the main program (see Fig.).
2.3 Subprograms with parameter transfer

Programming

The parameters relevant for parameter transfer must be listed at the beginning of the subprogram with their type and name.

**Parameter transfer call-by-value**

PROC PROGRAM_NAME(VARIABLE_TYPE1 VARIABLE1, VARIABLE_TYPE2 VARIABLE2,...)

Example:

PROC CONTOUR(REAL LENGTH, REAL WIDTH)

**Parameter transfer call-by-reference, identification with vocabulary word VAR**

PROC PROGRAM_NAME(VARIABLE_TYPE1 VARIABLE1, VARIABLE_TYPE2 VARIABLE2, ...)

Example:

PROC CONTOUR(VAR REAL LENGTH, VAR REAL WIDTH)

**Array transfer with call-by-reference, identification with vocabulary word VAR**

PROC PROGRAM_NAME(VAR VARIABLE_TYPE1 ARRAY_NAME1[array size], VAR VARIABLE_TYPE2 ARRAY_NAME2[array size], VAR VARIABLE_TYPE3 ARRAY_NAME3[array size1, array size2], VAR VARIABLE_TYPE4 ARRAY_NAME4[,], VAR VARIABLE_TYPE5 ARRAY_NAME5 [,array size])

Example:

PROC PALLET (VAR INT ARRAY[,10])

**Additional notes**

The definition statement with PROC must be programmed in a separate NC block. A maximum of 127 parameters can be declared for parameter transfer.
Array definition
The following applies to the definition of the formal parameters:
With two-dimensional arrays the number of fields in the first dimension does not need to be specified, but the comma must be written.
Example:
VAR REAL ARRAY[,5]

With certain array dimensions it is possible to process subprograms with arrays of variable length. However, when defining the variables you must define how many elements it is to contain.

See the Programming Guide "Advanced" for an explanation of array definition.

Programming example
Programming with variable array dimensions

%_N_DRILLING_PLATE_MPF
DEF REAL TABLE[100,2]
EXTERN DRILLING_PATTERN
(VAR REAL[,2],INT)
TABLE[0.0]=-17.5
TABLE[99.1]=45
DRILLING_PATTERN(TABLE,100)
M30

Creating a drilling pattern using the position table of variable dimension passed

%_N_DRILLING_PATTERN_SPF
PROC DRILLING_PATTERN(VAR REAL ARRAY[,2],-> INT NUMBER)
DEF INT COUNT
STEP: G1 X=ARRAY[COUNT,0]->
-> Y=ARRAY[COUNT,1] F100
Z=IC(-5)
Z=IC(5)
COUNT=COUNT+1
IF COUNT<NUMBER GOTOB STEP
RET

End of subprogram
2.4 Calling subprograms: L or EXTERN

Subprogram call without parameter transfer
In the main program you call the subprogram either with address L and the subprogram number or by specifying the program name.

Example:
N10 L47 or
N10 SPIGOT_2

Subprogram with parameter transfer
 declares with EXTERN
Subprograms with parameter transfer must be listed with EXTERN in the main program before they are called, e.g. at the beginning of the program. The name of the subprogram and the variable types are declared in the sequence in which they are transferred.

You only have to specify EXTERN if the subprogram is in the workpiece or in the global subprogram directory.
You do not have to declare cycles as EXTERN.

EXTERN statement
EXTERN NAME(TYP1, TYP2, TYP3, ...) or
EXTERN NAME(VAR TYP1, VAR TYP2, ...)

Example:
N10 EXTERN BORDER(REAL, REAL, REAL)...
N40 BORDER(15.3,20.2,5)

N10 Declaration of the subprogram, N40 Subprogram call with parameter transfer.
Subprogram call with parameter transfer

In the main program you call the subprogram by specifying the program name and parameter transfer. When transferring parameters you can transfer variables or values directly (not for VAR parameters).

Example:
N10 DEF REAL LENGTH, WIDTH, DEPTH
N20 ...
N30 LENGTH=15.3 WIDTH=20.2 DEPTH=5
N40 BORDER(LENGTH, WIDTH, DEPTH)
or
N40 BORDER(15.3, 20.2, 5)

Subprogram definition must match subprogram call

Both the variable types and the sequence of transfer must match the definitions declared under PROC in the subprogram name. The parameter names can be different in the main program and subprograms.

Example:
Definition in the subprogram:

PROC BORDER(REAL LENGTH, REAL WIDTH, REAL DEPTH)

Call in the main program:

N30 BORDER(LENGTH, WIDTH, DEPTH)
Incomplete parameter transfer

In a subprogram call only mandatory values and parameters can be omitted. In this case, the parameter in question is assigned the value zero in the subprogram.

The comma must always be written to indicate the sequence. If the parameters are at the end of the sequence you can omit the comma as well.

Back to the last example:

```
N30 LENGTH=15.3 WIDTH=20.2 WIDTH=5
N40 BORDER(15.3, 20.2, 5)
```

The mean value 20.2 was omitted here.

Note

*The current parameter of type AXIS must not be omitted.*

**VAR parameters must be passed on completely.**

SW 4.4 and higher:

With incomplete parameter transfer, it is possible to tell by the system variable \$P\_SUBPAR[\(i\)] whether the transfer parameter was programmed for subprograms or not.

The system variable contains as argument \((i)\) the number of the transfer parameter.

The system variable \$P\_SUBPAR returns

- **TRUE**, if the transfer parameter was programmed
- **FALSE**, if no value was set as transfer parameter.

If an impermissible parameter number was specified, parts program processing is aborted with alarm output.
### Example:

**Subprogram**

```plaintext
PROC SUB1 (INT VAR1, DOUBLE VAR2)

IF $P_SUBPAR[1]==TRUE
   ;Parameter VAR1 was not
   ;in the subprogram call
ELSE
   ;Parameter VAR1 was not
   ;programmed in the subprogram call
   ;and was preset by the system
   ;with default value 0
ENDIF
IF $P_SUBPAR[2]==TRUE
   ;Parameter VAR2 was not
   ;in the subprogram call
ELSE
   ;Parameter VAR2 was not
   ;programmed in the subprogram call
   ;and was preset by the system
   ;with default value 0.0
ENDIF
;Parameter 3 is not defined
IF $P_SUBPAR[3]==TRUE -> Alarm 17020
M17
```

### Calling the main program as a subprogram

A main program can also be called as subprogram. The end of program M2 or M30 set in the main program is evaluated as M17 in this case (end of program with return to the calling program).

Program the call by specifying the program name.

**Example:**

- N10 MPF739
- N10 SHAFT3

A subprogram can also be started as a main program.
2.5 Parameterizable subprogram return (SW 6.4 and higher)

Programming

Parameterizable subprogram return with the relevant parameters

RET (<block number/label>, <block after block with block number/label>,
     <number of return levels>), <return to beginning of program>)

RET (<block number/label>, <   >, <   >)

RET (, , <number of return levels>,
     <return to beginning of program>)

Subprogram return over two or more levels (jump back the specified number of levels).

Explanation

1st parameter: Block number or label as STRING (constant or variable) of the block at which to resume execution. Execution is resumed in the calling program at the block with the "Block number/label".

2nd parameter of type INTEGER
If the value is greater than 0, execution is resumed at "Block number/label". If the value is equal to 0, the subprogram return goes to the block with <block number/label>.

3rd parameter of type INTEGER with the permissible values 1 to 11.
Value = 1: The program is resumed in the current program level –1 (like RET without parameters).
Value = 2: The program is resumed in the current program level –2, skipping one level, etc.

4th parameter of type BOOL
Value 1 or 0.
Value = 1 If the return goes to the main program and ISO dialect mode is active there, execution will be resumed at the beginning of the program.
Function

Usually, a RET or M17 end of subprogram returns to the calling program and execution of the parts program continues with the lines following the subprogram call. However, some applications may require program resumption at another position:

- Continuation of execution after call-up of the cutting cycles in ISO dialect mode, after the contour definition.
- Return to main program from any subprogram level (even after ASUB) for error handling.
- Return over two or more program levels for special applications in compile cycles and in ISO dialect mode.

The parameterizable command RET can fulfill these requirements with 4 parameters:

1. <block number/label>
2. <block after block with block number/label>
3. <number of return levels>
4. <return to beginning of program>

1. **<block number/label>**

   Execution is resumed in the calling program (main program) at the block with the <block number/label>.
2. <block after block with block number/label>
The subprogram return goes back to the block with <block number/label>.

3. <number of return levels>
The program is resumed in the current program level minus <number of return levels>.
2.5 Parameterizable subprogram return (SW 6.4 and higher)

Impermissible return levels
Programming a number of return levels with
- a negative value or
- a value greater than the currently active program level −1 (max. 11),
will output Alarm 14091 with parameter 5.

Return with SAVE instructions
On return over two or more program levels, the SAVE instructions of each program level are evaluated.

Modal subprogram active on return
If a modal subprogram is active on a return over two or more program levels and if the deselection command MCALL is programmed for the modal subprogram in one of the skipped subprograms, the modal subprogram will remain active.

The user must always make sure that execution continues with the correct modal settings on return over two or more program levels.
This is done, for example, by programming an appropriate main block.

Programming example 1
Error handling: Resumption in main program after ASUB execution

| N10010 CALL "UP1"                   | ; Program level 0 main program |
| N11000 PROC UP1                     | ; Program level 1 |
| N11010 CALL "UP2"                   | ; Program level 2 |
| N12000 PROC UP2                     | ; Program level 2 (ASUB execution) |
| ... RET(“N10900”, , ...)            | ; Program level 3 |
| N19100 RET(N10900, ,$P_STACK)       | ; Subprogram return |
| N10900                              | ; Resumption in main program |
| N10910 MCALL                         | ; Deactivate modal subprogram |
| N10920 G0 G60 G40 M5                | ; Correct further modal settings |

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
2.6 Subprogram with program repetition: P

Program repetition, P

If you want to execute a subprogram several times in succession, you can program the required number of program repetitions in the block in the subprogram call under address P.

Example:

```
N40 BORDER P3
```

The subprogram Border must be executed three times in succession.

Value range:
P: 1…9999

The following applies to every subprogram call:

The subprogram call must always be programmed in a separate NC block.

Subprogram call with program repetition and parameter transfer

Parameters are only transferred during the program call or the first pass. The parameters remain unchanged for the repetitions.

If you want to change the parameters in the program repetitions you must define declarations in the subprograms.
2.7 Modal subprogram: MCALL

Modal subprogram call, MCALL

With this function the subprogram is automatically called and executed after every block with path motion. In this way you can automate the calling of subprograms that are to be executed at different positions on the workpiece. For example, for drilling patterns.

Examples:

```
N10 G0 X0 Y0
N20 MCALL L70
N30 X10 Y10
N40 X50 Y50
```

In blocks N30 to N40, the program position is approached and subprogram L70 is executed.

```
N10 G0 X0 Y0
N20 MCALL L70
N30 L80
```

In this example, the following NC blocks with programmed path axes are stored in subprogram L80. L70 is called by L80.

⚠️ In a program run, only one MCALL call can apply at any one time. Parameters are only passed once with an MCALL.

In the following situations the modal subprogram is also called without motion programming:
- When programming the addresses S and F if G0 or G1 is active.
- G0/G1 is on its own in the block or was programmed with other G codes.

Deactivating the modal subprogram call

With MCALL without a subprogram call or by programming a new modal subprogram call for a new subprogram.
2.8 Calling the subprogram indirectly: CALL

**Programming**

CALL <progname>

**Explanation**

CALL

<progname>

Vocabulary word for indirect subprogram call

Variable or constant of type string

Name of the program containing the program section to run

**Indirect subprogram call, CALL**

Depending on the prevailing conditions at a particular point in the program, different subprograms can be called.

The name of the subprogram is stored in a variable of type STRING. The subprogram call is issued with CALL and the variable name.

*The indirect subprogram call is only possible for subprograms without parameter transfer.*

For direct calling of the subprogram, store the name in a string constant.

Example:

**Direct call with string constant:**

CALL "/_N_WCS_DIR/_N_SUBPROG_WPD/_N_PART1_SPF"

**Indirect call via variable:**

DEF STRING[100] PROGNAME
PROGNAME="/_N_WCS_DIR/_N_SUBPROG_WPD/_N_PART1_SPF"
CALL PROGNAME

The subprogram PART1 is assigned the variable PROGNAME. With CALL and the path name you can call the subprogram indirectly.
2.9 Repeating program sections with indirect programming (SW 6.4 and higher)

Programming

CALL <progoname> BLOCK <startlabel> TO <endlabel>
CALL BLOCK <startlabel> TO <endlabel>

Explanation

CALL
Vocabulary word for indirect subprogram call

<progoname> (optional)
Variable or constant of type string, name of the program containing the program section to run.
If no <progoname> is programmed, the program section with <startlabel> <endlabel> in the current program is searched for and run.

BLOCK ... TO ...
Vocabulary word for indirect program section repetition

<startlabel> <endlabel>
Variable or constant of type string
Refers to the beginning or end of the program section to run

Function

CALL is used to call up subprogram indirectly in which the program section repetitions defined with BLOCK are run according to the start label and end label.

Programming example

DEF STRING[20] STARTLABEL, ENDLABEL
STARTLABEL = "LABEL_1"
ENDLABEL = "LABEL_2"
...
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDLABEL ...
M17
PROC CONTUR_1 ...
LABEL_1 ; Beginning of program section repetition
N1000 G1 ...  
LABEL_2 ; End of program section repetition
2.10 Calling up a program in ISO language indirectly: ISOCALL

Programming

ISOCALL <programe>

Explanation

ISOCALL

Subprogram call with which the ISO mode set in the machine data is activated

Variable or constant of type string

Name of the program in ISO language.

Function

The indirect program call ISOCALL is used to call up
a program in ISO language. The ISO mode set in the
machine data is activated

At the end of the program, the original mode is
reactivated. If no ISO mode is set in the machine
data, the subprogram is called in Siemens mode.

For more information about ISO mode, see
/FBFA/, "Description of Functions ISO Dialects"

Example:

Calling up a contour from ISO mode with cycle
programming:

%_N_0122_SPF
N1010 G1 X10 Z20
N1020 X30 R5
N1030 Z50 C10
N1040 X50
N1050 M99

N0010 DEF STRING[5] PROGNAME = "0122"
...
N2000 R11 = $AA_IW[X]
N2010 ISOCALL PROGNAME
N2020 R10 = R10+1
N2300 ...
N2400 M30

Contour description in ISO mode

Siemens parts program (cycle)

Run program 0122.spf in ISO mode
2.11 Calling subprogram with path specification and parameters PCALL

**Programming**

Subprogram call with the absolute path and parameter transfer

PCALL <path/progname>(parameter 1, ..., parameter n)

**Explanation**

PCALL

Vocabulary word for subprogram call with absolute path name

<path name>

Absolute path name beginning"\\", including subprogram names
If no absolute path name is specified, PCALL behaves like a standard subprogram call with a program identifier. The program identifier is written without the leading _N_ and without an extension
If you want the program name to be programmed with the leading _N_ and the extension, you must declare it explicitly with the leading _N_ and the extension as Extern.
Current parameters in accordance with the PROC statement of the subprogram

**Parameters 1 to n**

**Function**

With PCALL you can call subprograms with the absolute path and parameter transfer.

Example:

PCALL/_N_WCS_DIR/_N_SHAFT_WPD/SHAFT(parameter1, parameter2, ...)
2.12 Extending a search path for subprogram calls with CALLPATH  
(SW 6.4 and higher)

**Programming**

Adding subprograms stored outside the existing NCK file system to the existing NCK file system.

CALLPATH <path name>

**Explanation**

CALLPATH  

Constant or variable of type string contains the absolute path of a directory beginning with "/" to extend the search path. The path must be specified complete with prefixes and suffixes. (e.g.: /_N_WKS_DIR/_N_WST_WPD)  

If <path name> contains the empty string or if CALLPATH is called without parameters, the search path instruction will be reset. The maximum path length is 128 bytes.

**Function**

The CALLPATH command is used to extend the search path for subprogram calls. That allows you to call subprograms from a non-selected workpiece directory without specifying the complete absolute path name of the subprogram. Search path extension comes before the user cycle entry. (_N_CUS-DIR).
Example:
CALLPATH ("/N_WKS_DIR/N_MYWPD_WPD")

That sets this search path (position 5 is new):
1. current directory/ subprogram identifier
2. current directory/ subprogram identifier_SPF
3. current directory/ subprogram identifier_MPF
4. /N_SPF_DIR/ subprogram identifier_SPF
5. /N_WKS_DIR/N_MYWPD/ subprogram identifier_SPF
6. N_CUS_DIR/N_MYWPD/ subprogram identifier_SPF
7. /N_CMA_DIR/ subprogram identifier_SPF
8. /N_CST_DIR/ subprogram identifier_SPF

Deselection of the search path extension
The search path extension is deselected by the following results:
- CALLPATH with empty string
- CALLPATH without parameters
- End of parts program
- RESET

Additional notes
- CALLPATH check whether the programmed path name really exists. An error aborts program execution with correction block alarm 14009.
- CALLPATH call also be programmed in INI files. Then it applies for the duration of execution of the INI file (WPD INI file or initialization program for NC active data, e.g. Frames in the 1st channel _N_CH1_UFR_INI). The initialization program is then reset again.
2.13 Suppress current block display: DISPLOF

Programming

PROC ... DISPLOF

Function

With DISPLOF the current block display is suppressed for a subprogram. DISPLOF is placed at the end of the PROC statement. Instead of the current block, the call of the cycle or the subprogram is displayed.

By default the block display is activated. Deactivation of block display with DISPLOF applies until the return from the subprogram or end of program. If further subprograms are called from the subprogram with the DISPLOF attribute, the current block display is suppressed in these as well. If a subprogram with suppressed block display is interrupted by an asynchronized subprogram, the blocks of the current subprogram are displayed.

Programming example

Suppress current block display in the cycle

%_N_CYCLE_SPF
;$PATH=/_N_CUS_DIR
PROC CYCLE (AXIS TOMOV, REAL POSITION) SAVE DISPLOF

; Suppress current block display
; Now the cycle call is displayed as the current block
; E.g.: CYCLE(X, 100.0)
DEF REAL DIFF

; Cycle contents
G01 ...
...
RET

; Subprogram return, the following block of the calling program is displayed again
2.14 Single block suppression: SBLOF, SBLON (SW 4.3 and higher)

Programming

PROC ... SBLOF
SBLON

; The command can be programmed in a PROC block or in a separate block

; The command must be programmed in a separate block

Explanation

SBLOF  Deactivate single block
SBLON  Reactivate single block

Function

Program-specific single block suppression

With all single block types the programs marked with SBLOF are executed in their entirety like one block. SBLOF is written in the PROC line and is valid until the end of the subprogram or until it is aborted.

SBLOF is also valid in the called subprograms.

Example:
PROC EXAMPLE SBLOF
G1 X10
RET

Single block suppression in the program

SBLOF can be alone in a block. From this block onwards, the single block mode is deactivated until

• the next SBLON or
• until the end of the active subprogram level.

Example:
N10  G1  X100  F1000
N20  SBLOF
N30  Y20
N40  M100
N50  R10=90
N60  SBLON
N70  M110
N80  ...

Deactivate single block
Reactivate single block

The range between N20 and N60 is executed in single block mode as one step.
Single block disable for asynchronized subprograms

To run an ASUB in single block mode in one step, the ASUB must contain a PROC instruction with SBLOF.
This also applies to the function "editable system ASUB" in MD 11610: ASUB_EDITABLE.

Example of "editable system ASUB":
N10 PROC ASUB1 SBLOF DISPLOF
N20 IF $AC_ASUB=='H200'
N30 RET
N40 ELSE
N50 REPOSA
N60 ENDF

Program control in single block mode

The single block function allows the user to run a parts program block by block. The single block function has the following settings:
- SBL1: IPO single block with stop after each machine function block.
- SBL2: Single block with stop after each block.
- SBL3: Hold in the cycle

Single block suppression for program nesting

If "single block off" (SBLOF) is active in a subprogram, execution halts at the M17 instruction. That prevents the next block in the calling program from already running.
If single block suppression is deactivated in a subprogram with SBLON/SBLOF (independently of the attribute SBLOF), execution stops after the next block of the calling program.
If that is not wanted, SBLON must be programmed in the subprogram before the return (M17).
Execution does not stop on a return to a higher-level program with RET.
Supplementary conditions

- Display of the current block can be suppressed in cycles by means of DISPLOF.
- If DISPLOF is programmed together with SBLOF, then the cycle call is still displayed in single block stops within a cycle.
- If MD 10702: IGNORE_SINGLEBLOCK_MASK suppressed the single block stop in the system ASUB or user ASUB with Bit0 = 1 or Bit1 = 1, you can reactivate the single block stop by programming SBLON in ASUB.
- MD 20117: IGNORE_SINGLEBLOCK_ASUB suppresses the single block stop in the user ASUB and cannot be reactivated any more by programming SBLON.

- By selecting SBL3 you can suppress the SBLOF command.
- **SW 6.4 and higher**
  Ignore single block stop in single block type 2. 
  Single block type 2 (SBL2) does not stop in the SBLON block, if Bit12 = 1 is set in MD 10702: IGNORE_SINGLEBLOCK_MASK.

For more information about block display with/without single block suppression, see /FB/, K1 BAG, Channel, Program control "single block"
### Programming example 1

**Cycle is to act as a command for programmer**

#### Main program

```
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30
```

#### Program cycle1

```
N100 PROC CYCLE1 DISPLOF SBLOF ; Suppress single block
N110 R10=3*SIN(R20)+5
N120 IF (R11 <= 0)
N130    SETAL(61000)
N140 ENDIF
N150 G1 G91 Z=R10 F=R11
N160 RET
```

The cycle CYCLE1 is executed as one step when single block is active.

### Programming example 2

**An ASUB run from the PLC for activating modified zero offsets and tool offsets should not be visible.**

```
N100 PROC NV SBLOF DISPLOF
N110 CASE $P_UIFRNUM OF 0 GOTOF _G500
    -->1 GOTOF _G54 2 GOTOF _G55 3
    -->GOTO _G56 4 GOTOF _G57
    -->DEFAULT GOTOF END
N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO
N130 RET
N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO
N150 RET
N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO
N170 RET
N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO
N190 RET
N200 END: D=$P_TOOL T=$P_TOOLNO
N210 RET
```
Programming example 3

No stop with MD 10702: IGNORE_SINGLEBLOCK_MASK, Bit 12 = 1

In single block type SBL2 (stop at each parts program line) in the SBLON instruction

;SBL2 is active

;$MN_IGNORE_SINGLEBLOCK_MASK = `H1000` ; Set MD 10702: Bit 12 = 1

N10  G0  X0 ; Stop at this parts program line
N20  X10 ; Stop at this parts program line
N30  CYCLE ; Traversing block generated by the cycle

PROC CYCLE SBLOF ; Suppress single block stop

N100  R0 = 1

N110  SBLON ; Because MD 10702: Bit 12 = 1, no stop

N120  X1 ; Stop at this parts program line
N140  SBLOF
N150  R0 = 2
RET

N50  G90  X20 ; Stop at this parts program line
M30
### Programming example 4

Single block suppression for program nesting

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>X0 F1000</td>
</tr>
<tr>
<td>N20</td>
<td>UP1(0)</td>
</tr>
<tr>
<td></td>
<td>PROC_UP1(INT__NR) SBLOF</td>
</tr>
<tr>
<td></td>
<td>N100 X10</td>
</tr>
<tr>
<td></td>
<td>N110 UP2(0)</td>
</tr>
<tr>
<td></td>
<td>PROC_UP2(INT__NR)</td>
</tr>
<tr>
<td></td>
<td>N200 X20</td>
</tr>
<tr>
<td></td>
<td>N210 SBLON</td>
</tr>
<tr>
<td></td>
<td>N220 X22</td>
</tr>
<tr>
<td></td>
<td>N230 UP3(0)</td>
</tr>
<tr>
<td></td>
<td>PROC_UP3(INT__NR)</td>
</tr>
<tr>
<td></td>
<td>N302 SBLOF</td>
</tr>
<tr>
<td></td>
<td>N300 X30</td>
</tr>
<tr>
<td></td>
<td>N310 SBLON</td>
</tr>
<tr>
<td></td>
<td>N320 X32</td>
</tr>
<tr>
<td></td>
<td>N330 SBLOF</td>
</tr>
<tr>
<td></td>
<td>N340 X34</td>
</tr>
<tr>
<td></td>
<td>N350 M17</td>
</tr>
<tr>
<td></td>
<td>N240 X24</td>
</tr>
<tr>
<td></td>
<td>N250 M17</td>
</tr>
<tr>
<td></td>
<td>N120 X12</td>
</tr>
<tr>
<td></td>
<td>N130 M17</td>
</tr>
<tr>
<td></td>
<td>N30 X0</td>
</tr>
<tr>
<td></td>
<td>N40 M30</td>
</tr>
</tbody>
</table>
### 2.15 Executing external subprogram: EXTCALL (SW 4.2 and higher)

#### Programming

```plaintext
EXTCALL (<path/program name>)
```

#### Explanation

- **EXTCALL**: Keyword for subprogram call
- `<path/program name>`: Constant/variable of type STRING. An absolute path name or program name can be specified. The program name is written with/without the leading `N_` and without an extension. An extension can be appended to the program name using the `"` character.

**Example:**

```plaintext
EXTCALL ("/N_WK_DIR/N_SHAFT_WPD/N_SHAFT_SPF") or EXTCALL ("SHAFT")
```

#### Function

- **EXTCALL** can be used to reload a program from the HMI in "Processing from external source" mode. All programs that can be accessed via the directory structure of HMI can be reloaded and run.

**External program path**

SD 42700: `EXT_PROG_PATH` permits flexible setting of the call path. SD 42700 contains a path definition that builds the absolute path name of the program to be called in conjunction with the programmed subprogram identifier.

**Call of an external subprogram**

An external subprogram is called up with parts program command `EXTCALL`.

The

- subprogram names programmed with `EXTCALL` and
- setting data SD 42700: `EXT_PROG_PATH`
result in the program path for the external subprogram call by character concatenation of:

- the content of SD 42700: EXT_PROG_PATH (e.g. */_N_WKS_DIR/_N_WKST1_WPD)
- the character "/" as the separator (if a path was specified with SD 42700: EXT_PROG_PATH)
- the subprogram path or subprogram identifier specified with EXTCALL.

SD 42700: EXT_PROG_PATH has a blank as its default. If the external subprogram is called without an absolute path name, the same search path is executed on the HMI Advanced as for calling a subprogram from NCK memory.

1. current directory/subprogram identifier
2. current directory/subprogram identifier_SPF
3. current directory/subprogram identifier_MPF
4. */_N_SPF_DIR/subprogram identifier_SPF
5. */_N_CUS_DIR/subprogram identifier_SPF
6. */_N_CMA_DIR/subprogram identifier_SPF
7. */_N_CST_DIR/subprogram identifier_SPF

"current directory": stands for the directory in which the main program was selected.
"subprogram identifier": stands for the subprogram name programmed with EXTCALL.

Adjustable load memory (FIFO buffer)
A load memory is required in the NCK in order to process a program in "Execution from external" mode (main program or subprogram). The default setting for the size of the load memory is 30 Kbytes. MD 18360: MM_EXT_PROG_BUFFER_SIZE sets the size of the reload buffer. MD 18362: MM_EXT_PROG_BUFFER_NUM sets the number of reload buffers. One reload buffer must be set for each program (main program or subprogram) to run concurrently in "Processing from external source" mode.
2.15 Executing external subprogram: EXTCALL (SW 4.2 and higher)

Programming examples

1. The program to be reloaded is located on the local hard disk of HMI Advanced:

Setting data SD 42700: EXT_PROG_PATH contains the following path: "/_N_WKS_DIR/_N_WST1".
The main program _N_MAIN_MPF is in the user memory and selected.

N10 PROC MAIN
N20 ...
N30 EXTCALL "ROUGHING" ; Call of external subprogram ROUGHING
N40 ...
N50 M30

Subprogram "ROUGHING" (located in the HMI Advanced directory structure under workpieces -> WST1):

N10 PROC ROUGHING
N20 G1 F1000
N30 X=... Y=... Z=...
N40 ...
N90 M17

2. The program to be reloaded is located on the network drive or ATA card of HMI

EXTCALL Windows path

Call for network drive (HMI Embedded or Advanced)
EXTCALL \R4711\workpieces\contour1.spf

Call for ATA card (HMI Embedded), e.g.
EXTCALL C:\workpieces\contour2.spf

For HMI Embedded, an absolute path must always be specified.

For more information about operation, see
/BEM/ HMI Embedded
/BAD/ HMI Advanced
2.15 Executing external subprogram: EXTCALL (SW 4.2 and higher)

**Additional notes**

External subprograms are not permitted to include jump commands such as GOTOF, GOTOB, CASE, FOR, LOOP, WHILE or REPEAT.

Subprogram calls – even nested EXTCALL calls are possible.

**SW 6.3 and higher**

IF-ELSE-ENDIF constructions are possible.

**POWER ON, RESET**

Reset and power ON cause external subprogram calls to be interrupted and the associated load memory to be erased.

For more information about "Processing from external source", see:
/FB/ K1, BAG, Channel, Program control
2.16 Subprogram call with M/T function

**Function**

The T/M function can be replaced with a subprogram call by making the appropriate setting in the machine data, for example, for calling the tool change routine. At block search subprogram calls with M/T functions behave like standard subprogram calls.

For more information about "Subprogram call with M/T functions", see:
/FB/ K1, BAG, Channel, Program control

### Example 1: Tool change with M6

M function M6 is replaced by tool change routine TC_UP_M6

<table>
<thead>
<tr>
<th>line</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>PROC ROUGHING3</td>
</tr>
<tr>
<td>N20</td>
<td>G1 F1000</td>
</tr>
<tr>
<td>N30</td>
<td>X=... Y=... Z=...</td>
</tr>
<tr>
<td>N40</td>
<td>T1234 M6 ;</td>
</tr>
<tr>
<td></td>
<td>; Call TC_UP_M6</td>
</tr>
<tr>
<td>M30</td>
<td></td>
</tr>
</tbody>
</table>

Associated subprogram TC_UP_M6:

<table>
<thead>
<tr>
<th>line</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N110</td>
<td>PROC TC_UP_M6</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>N130</td>
<td>G53 D0 G0 X=... Y=... Z=... ;</td>
</tr>
<tr>
<td></td>
<td>; Approach tool change point</td>
</tr>
<tr>
<td>N140</td>
<td>M6 ;</td>
</tr>
<tr>
<td></td>
<td>; Execute tool change</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>N190</td>
<td>M17</td>
</tr>
</tbody>
</table>

### Example 2: Tool change with T function programming

T function is replaced by tool change routine TC_UP_T

<table>
<thead>
<tr>
<th>line</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>PROC ROUGHING4</td>
</tr>
<tr>
<td>N20</td>
<td>G1 F1000</td>
</tr>
<tr>
<td>N30</td>
<td>X=... Y=... Z=...</td>
</tr>
<tr>
<td>N40</td>
<td>T1234 ;</td>
</tr>
<tr>
<td></td>
<td>; Call TC_UP_T</td>
</tr>
<tr>
<td>M30</td>
<td></td>
</tr>
</tbody>
</table>
Associated subprogram TC_UP_T:

```
N310 PROC TC_UP_T
...
N330 IF $C_T_PROG == 1
N340 G53 D0 G0 X=... Y=... Z=... ; Approach tool change point
N350 T=$C_T ; Execute tool change
N360 ENDFI
...
N390 M17
```

Extension of T function substitution

As from SW 6.4, T function substitution is extended to permit setting in machine data whether with programming of both:

**D numbers or DL numbers and T numbers**

- in one block, D or DL will be passed as parameters to the T substitution cycle as predefined (default) or
- run before the T substitution cycle call.

MD 10719: T_NO_FCT_CYCLE_MODE sets parameterization of the T function substitution as follows

Value 0: the D or DL number is passed to the cycle, as previously, (default).
Value 1: the D or DL number is calculated directly in the block.

This function is only active if the tool change was configured with an M function (MD 22550: TOOL_CHANGE_MODE = 1), otherwise the D or DL values are always passed.
2.17 Cycles: Setting parameters for user cycles

Files and paths

Explanation

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cov.com</td>
<td>Overview of cycles</td>
</tr>
<tr>
<td>uc.com</td>
<td>Cycle call description</td>
</tr>
</tbody>
</table>

Function

Customized cycles can be parameterized with these files.

Sequence

The cov.com file is included with the standard cycles at delivery and is to be expanded accordingly. The uc.com file is to be created by the user.

Both files are to be loaded in the passive file system in the "User cycles" directory (or must be given the appropriate path specification in the program: 

```c
;PATH=/_N_CUS_DIR
```

Adaptation of cov.com – Overview of cycles

The cov.com file supplied with the standard cycles has the following structure:

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_COV_COM</td>
<td>File name</td>
</tr>
<tr>
<td>;PATH=/_N_CST_DIR</td>
<td>Path specification</td>
</tr>
<tr>
<td>;Vxxx 11.12.95 Sca cycle overview</td>
<td>Comment line</td>
</tr>
<tr>
<td>C1(CYCLE81) drilling, centering</td>
<td>Call for 1st cycle</td>
</tr>
<tr>
<td>C2(CYCLE82) Boring, counterboring</td>
<td>Call for 2nd cycle</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>C24(CYCLE98) Chaining of threads</td>
<td>Call for last cycle</td>
</tr>
<tr>
<td>M17</td>
<td>End of file</td>
</tr>
</tbody>
</table>
For each newly added cycle a line must be added with the following syntax:

```
C<Number> (Cycle name) comment text
```

Number: Any integer, must not have been used in the file before;
Cycle name: The program name of the cycle to be included
Comment text: Optionally a comment text for the cycle

**Example:**

```
C25 (MY_CYCLE_1) usercycle_1
C26 (SPECIAL CYCLE)
```

**Example of uc.com file**

**User cycle description**

The explanation is based on the continuation of the example:

For the following two cycles a cycle parameterization is to be newly created:

**PROC MY_CYCLE_1 (REAL PAR1, INT PAR2, CHAR PAR3, STRING[10] PAR4)**

```
;The cycle has the following transfer parameters:

;PAR1: Real value in range -1000.001 <= PAR2 <= 123.456, default with 100
;PAR2: Positive integer value between 0 <= PAR3 <= 999999, Default with 0
;PAR3: 1 ASCII character
;PAR4: String of length 10 for a subprogram name

...
M17
```

**PROC SPECIALCYCLE (REAL VALUE1, INT VALUE2)**

```
;The cycle has the following transfer parameters:

;VALUE1: Real value without value range limitation and default
;VALUE2: Integer value without value range limitation and default

...
M17
```
Associated file uc.com

%_N_UC_COM

; $PATH=/_N_CUS_DIR

//C25(MY_CYCLE_1) usercycle_1
(R/-1000.001 123.456 / 100 /Parameter_2 of cycle)
(I/0 999999 / 1 / integer value)
(C//"A" / Character parameter)
(S///Subprogram name)

//C26(SPECIALCYCLE)
(R///Entire length)
(I/*123456/3/Machining type)
M17

Syntax description for the uc.com file –
user cycle description

Header line for each cycle:
as in the cov.com file preceded by "//"

//C<Number> (<Cycle name>) comment text

Example:
//C25(MY_CYCLE_1) usercycle_1
Line for description for each parameter:

\[(<\text{data type identifier}> / <\text{minimum value}> <\text{maximum value}> / <\text{default value}> / <\text{Comments}>)\]

**Data type identifier:**

- \(R\) for real
- \(I\) for integer
- \(C\) for character (1 character)
- \(S\) for string

**Minimum value, maximum value** (can be omitted)

Limitations of the entered values which are checked at input; values outside this range cannot be entered.

It is possible to specify an enumeration of values which can be operated via the toggle key; they are listed preceded by "\(*\)", other values are then not permissible.

Example:

\((I/*123456/1/Machining type)\)

There are no limits for string and character types;

**Default value** (can be omitted)

Value which is the default value in the corresponding screen when the cycle is called; it can be changed via operator input.

Comment

Text of up to 50 characters which is displayed in front of the parameter input field in the call screen for the cycle.
Display example for both cycles

Display screen for cycle MY_CYCLE_1

<table>
<thead>
<tr>
<th>Parameter 2 of the cycle</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer value</td>
<td>1</td>
</tr>
<tr>
<td>Character parameter</td>
<td></td>
</tr>
<tr>
<td>Subprograms</td>
<td></td>
</tr>
</tbody>
</table>

Display screen for cycle SPECIAL CYCLE

<table>
<thead>
<tr>
<th>Total length</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of machining</td>
<td>1</td>
</tr>
</tbody>
</table>

2.18 Macros. DEFINE...AS

What is a macro?
A macro is a sequence of individual instructions which have together been assigned a name of their own. G, M and H functions or L subprogram names can also be used as macros. When a macro is called during a program run, the instructions programmed under the program name are executed one after the other.

Use of macros
Sequences of instructions that recur are only programmed once as a macro in a separate macro module and once at the beginning of the program. The macro can then be called in any main program or subprogram and executed.
Programming

Macros are identified with the vocabulary word DEFINE...AS.

The macro definition is as follows:
DEFINE NAME AS <Instruction>

Example:
Macro definition:
DEFINE LINE AS G1 G94 F300

Call in the NC program:
N20  LINE X10 Y20

Activate macro

- **SW 4** and lower
  - Macros are active after control power ON.

- **SW 5** and higher
  - The macro is active when it is loaded into the NC ("Load" soft key).

Three-digit M/G function (SW 5 and higher)

- **SW 4** and lower
  - After a three-digit M function is programmed, alarm 12530 is issued.

- **SW 5** and higher
  - Supports programming of three-digit M and G functions.
    - Example:
      N20 DEFINE M100 AS M6
      N80 DEFINE M999 AS M6

Additional notes

Nesting of macros is not possible.

Two-digit H and L functions can be programmed.
2.18 Macros. DEFINE...AS

Programming example

Example of macro definitions.

```plaintext
DEFINE M6 AS L6
A subroutine is called at tool change to handle the necessary data transfer. The actual M function is output in the subprogram (e.g. M106).

DEFINE G81 AS DRILL(81)
Emulation of the DIN G function

DEFINE G33 AS M333 G333
During thread cutting synchronization is requested with the PLC. The original G function G33 was renamed to G333 by machine data so that the programming is identical for the user.
```

Example of a global macro file:

After reading the macro file into the control, activate the macros (see above). The macros can now be used in the parts program.

```plaintext
%_N_UMAC_DEF
;%PATH=_N_DEF_DIR; customer-specific macros
DEFINE PI AS 3.14
DEFINE TC1 AS M3 S1000
DEFINE M13 AS M3 M7 ;Spindle right, coolant on
DEFINE M14 AS M4 M7 ;Spindle left, coolant on
DEFINE M15 AS M5 M9 ;Spindle stop, coolant off
DEFINE M6 AS L6 ;Call tool change program
DEFINE G80 AS MCALL ;Deselect drilling cycle
M30
```

- Vocabulary words and reserved names must not be redefined with macros.
- Use of macros can significantly alter the control’s programming language! Therefore, exercise caution when using macros.
- Macros can also be declared in the NC program. Only identifiers are permissible as macro names. G function macros can only be defined in the macro module globally for the entire control.
- With macros you can define any identifiers, G, M, H functions and L program names.
- Macro identifiers with 1 letter and 1 digit are permissible (FM-NC only).
## File and Program Management

1. Overview ........................................................................................................................................... 3-146
2. Program memory ............................................................................................................................... 3-147
3. User memory ...................................................................................................................................... 3-153
4. Defining user data ............................................................................................................................. 3-156
5. Defining protection levels for user data (GUD) ................................................................................... 3-160
6. Automatic activation of GUDs and MACs (SW 4.4 and higher) ...................................................... 3-162
7. Data-specific protection level change for machine and setting data .................................................. 3-164
   7.1 Change ........................................................................................................................................... 3-164
   7.2 Undoing a change ............................................................................................................................ 3-165
8. Changing attributes of NC language elements (SW 6.4 and higher) ............................................... 3-165
9. Structuring instruction SEFORM in the Step editor (SW 6.4 and higher) ....................................... 3-173
3.1 Overview

Memory structure

The memory structure available to the user is organized in two areas.

1. User memory
The user memory contains the current system and user data with which the control operates (active file system).
Example:
Active machine data, tool offset data, zero offsets.

2. Program memory
The files and programs are stored in the program memory and are thus permanently stored (passive file system).
Example:
Main programs and subprograms, macro definitions.
3.2 Program memory

Overview

Main programs and subprograms are stored in the main memory. A number of file types are also stored here temporarily and these can be transferred to the working memory as required (e.g. for initialization purposes on machining of a specific workpiece).
Directories

The following directories exist by default:

1. _N_DEF_DIR  Data modules and macro modules
2. _N_CST_DIR  Standard cycles
3. _N_CMA_DIR  Manufacturer cycles
4. _N_CUS_DIR  User cycles
5. _N_WKS_DIR  Workpieces
6. _N_SPF_DIR  Global subprograms
7. _N_MPF_DIR  Standard directory for main programs
8. _N_COM_DIR  Standard directory for comments

File types

The following file types can be stored in the main memory:

<table>
<thead>
<tr>
<th>name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_MPF</td>
<td>Main program</td>
</tr>
<tr>
<td>_SPF</td>
<td>Subprogram</td>
</tr>
<tr>
<td>_TEA</td>
<td>Machine data</td>
</tr>
<tr>
<td>_SEA</td>
<td>Setting data</td>
</tr>
<tr>
<td>_TOA</td>
<td>Tool offsets</td>
</tr>
<tr>
<td>_UFR</td>
<td>Zero offsets/frames</td>
</tr>
<tr>
<td>_INI</td>
<td>Initialization file</td>
</tr>
<tr>
<td>_GUD</td>
<td>Global user data</td>
</tr>
<tr>
<td>_RPA</td>
<td>R parameters</td>
</tr>
<tr>
<td>_COM</td>
<td>Comments</td>
</tr>
<tr>
<td>_DEF</td>
<td>Definitions for global user data and macros</td>
</tr>
</tbody>
</table>
Workpiece directory, _N_WCS_DIR
The workpiece directory exists in the standard setup of the program directory under the name _N_WCS_DIR.

The workpiece directory contains all the workpiece directories for the workpieces that you have programmed.

Workpiece directories, Identifier WPD
To make data and program handling more flexible certain data and programs can be grouped together or stored in individual workpiece directories.
A workpiece directory contains all files required for machining a workpiece.

These can be main programs, subprograms, any initialization programs and comment files.

Example:
Workpiece directory _N_SHAFT_WPD, created for workpiece SHAFT contains the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_N_SHAFT_MPF</td>
<td>Main program</td>
</tr>
<tr>
<td>_N_PART2_MPF</td>
<td>Main program</td>
</tr>
<tr>
<td>_N_PART1_SPF</td>
<td>Subprogram</td>
</tr>
<tr>
<td>_N_PART2_SPF</td>
<td>Subprogram</td>
</tr>
<tr>
<td>_N_SHAFT_INI</td>
<td>General initialization program for the data of the workpiece</td>
</tr>
<tr>
<td>_N_SHAFT_SEA</td>
<td>Setting data initialization program</td>
</tr>
<tr>
<td>_N_PART2_INI</td>
<td>General initialization program for the Part 2 program</td>
</tr>
<tr>
<td>_N_PART2_UFR</td>
<td>Initialization program for the frame data for the Part 2 program</td>
</tr>
<tr>
<td>_N_SHAFT_COM</td>
<td>Comment file</td>
</tr>
</tbody>
</table>
Creating workpiece directories on an external PC

The steps described below are performed on an external data station.

Please refer to your Operator’s Guide for file and program management (from PC to control system) directly on the control.

;$PATH instruction
The destination path $PATH=... is specified within the second line of the file.

Example:
; $PATH=/_N_WCS_DIR/_N_SHAFT_WPD

The file is stored at the specified path.

Important
If the path is missing, files of file type SPF are stored in /_N_SPF_DIR, files with extension _INI in the working memory and all other files in /_N_MPF_DIR.

Example with path for the previous example SHAFT:
_/N_SHAFT_MPF is stored in
_/N_WKS_DIR/_N_SHAFT_WPD

%_N_SHAFT_MPF
; $PATH=/_N_WCS_DIR/_N_SHAFT_WPD
N10 G0 X... Z...

* M2

SHAFT:/_N_SHAFT_SPF is stored in
_/N_SPF_DIR

* %_N_SHAFT_SPF

* M17
**Select workpiece for machining**

A workpiece directory can be selected for execution in a channel.

If a main program with the **same name** or only a single main program (MPF) is stored in this directory, this is automatically selected for execution.

Example:

The workpiece directory

```
/_N_WCS_DIR/_N_SHAFT_WPD
```

contains the files

- `_N_SHAFT_SPF`
- `_N_SHAFT_MPF`

**SW 5 and higher (MMC 102/103 only):**

See "Operator's Guide" /BA/ Section on Job list and Selecting program for execution.

**Search path with subprogram call**

If the search path is not specified explicitly in the parts program when a subprogram (or initialization file) is called, the calling program searches in a fixed search path.

Example of subprogram call with absolute path specification:

```
CALL"/_N_CST_DIR/_N_CYCLE1_SPF"
```

Programs are usually called without specifying a path:

Example:

```
CYCLE1
```

**Search path sequence**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Current directory / <strong>name</strong></td>
</tr>
<tr>
<td>2.</td>
<td>Current directory / <strong>name</strong><em>SPF</em></td>
</tr>
<tr>
<td>3.</td>
<td>Current directory / <strong>name</strong><em>MPF</em></td>
</tr>
<tr>
<td>4.</td>
<td>/_/N_SPF_DIR / <strong>name</strong><em>SPF</em></td>
</tr>
<tr>
<td>5.</td>
<td>/_/N_CUS_DIR / <strong>name</strong><em>SPF</em></td>
</tr>
<tr>
<td>6.</td>
<td>/_/N_CMA_DIR / <strong>name</strong><em>SPF</em></td>
</tr>
<tr>
<td>7.</td>
<td>/_/N_CST_DIR / <strong>name</strong><em>SPF</em></td>
</tr>
</tbody>
</table>
Programming search paths for subprogram call (as from SW 6.4)

**CALLPATH command**

The search path can be extended with the parts program command `CALLPATH`.

Example:

```
CALLPATH("/_N_WKS_DIR/_N_MYWPDPWD")
```

The search path is stored in front of position 5 (user cycle) as programmed.

For further information about the programmable search path for subprogram calls with `CALLPATH`, see Section 2.12.
3.3 User memory

Initialization programs

These are programs with which the working memory data are initialized.

The following file types can be used for this:

- `name_TEA` Machine data
- `name_SEA` Setting data
- `name_TOA` Tool offsets
- `name_UFR` Zero offsets/frames
- `name_INI` Initialization files
- `name_GUD` Global user data
- `name_RPA` R parameters

Data areas

The data can be organized in different areas in which they are to apply. For example, a control can use several channels (not 810D CCU1, 840D NCU 571) and can usually use several axes. The following areas are available:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCK</td>
<td>NCK-specific data</td>
</tr>
<tr>
<td>CHn</td>
<td>Channel-specific data</td>
</tr>
<tr>
<td></td>
<td>(n specifies the channel number)</td>
</tr>
<tr>
<td>AXn</td>
<td>Axis-specific data (n specifies the number of the machine axis)</td>
</tr>
<tr>
<td>TO</td>
<td>Tool data</td>
</tr>
<tr>
<td>COMPLETE</td>
<td>All data</td>
</tr>
</tbody>
</table>
Generating an initialization program on an external PC

The data area identifier and the data type identifier can be used to determine the areas which are to be treated as a unit when the data are saved.

Example:

<table>
<thead>
<tr>
<th>Data Area Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_N_AX5_TEA_INI</td>
<td>Machine data for axis 5</td>
</tr>
<tr>
<td>_N_CH2_UFR_INI</td>
<td>Frames of channel 2</td>
</tr>
<tr>
<td>_N_COMPLETE_TEA_INI</td>
<td>All machine data</td>
</tr>
</tbody>
</table>

When the control is started up initially, a set of data is automatically loaded to ensure proper operation of the control.

Saving initialization programs

The files in the working memory can be saved on an external PC and read in again from there.

- The files are saved with COMPLETE.
- AnINI file: INITIAL can be created across all areas with _N_INITIAL_INI.

Loading initialization programs

INI programs can also be selected and called as parts programs if they only use the data of a single channel. It is thus also possible to initialize program-controlled data.

Information on file types is given in the Operator's Guide.
Procedure for multi-channel controls

CHANDATA (channel number) for several channels is only permitted in the file N_INITIAL_INI. N_INITIAL_INI is the installation file with which all data of the control is initialized.

Example:
```plaintext
%_N_INITIAL_INI
CHANDATA(1)
;Machine axis assignment channel 1
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
CHANDATA(2)
;Machine axis assignment channel 2
$MC_AXCONF_MACHAX_USED[0]=4
$MC_AXCONF_MACHAX_USED[1]=5
CHANDATA(1)
;axial machine data
;Exact stop window coarse:
$MA_STOP_LIMIT_COARSE[AX1]=0.2 ;Axis 1
$MA_STOP_LIMIT_COARSE[AX2]=0.2 ;Axis 2
;Exact stop window fine:
$MA_STOP_LIMIT_COARSE[AX1]=0.01 ;Axis 1
$MA_STOP_LIMIT_COARSE[AX1]=0.01 ;Axis 2
```

In the parts program, the CHANDATA instruction may only be used for the channel on which the NC program is running, i.e. the instruction can be used to protect NC programs from being executed accidentally on a different channel. Program processing is aborted if an error occurs.

Note
INI files in job lists do not contain any CHANDATA instructions.
3.4 Defining user data

Function

Definition of user data (GUD) implemented during start-up procedure.

The necessary machine data should be initialized accordingly. The user memory must be configured. All relevant machine data have as a component of their name GUD.

- **SW 5 and higher (01.99):**
  The user data (GUD) can be defined in the Services operating area. This means that lengthy reimporting of data backup (%_N_INITIAL_INI) is not necessary.
  The following applies:
  - Definition files that are on the hard disk are not active.
  - Definition files that are on the NC are always active.

Reserved module names

The following modules can be stored in the directory /_N_DEF_DIR:

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_N_SMAC_DEF</td>
<td>Contains macro definitions (Siemens, protection level 0)</td>
</tr>
<tr>
<td>_N_MMAC_DEF</td>
<td>Contains macro definitions (machine manufacturer, protection level 2)</td>
</tr>
<tr>
<td>_N_UMAC_DEF</td>
<td>Contains macro definitions (user, protection level 3)</td>
</tr>
<tr>
<td>_N_SGUD_DEF</td>
<td>Contains definitions for global data (Siemens, protection level 0)</td>
</tr>
<tr>
<td>_N_MGUD_DEF</td>
<td>Contains definitions for global data (machine manufacturer, protection level 2)</td>
</tr>
<tr>
<td>_N_UGUD_DEF</td>
<td>Contains definitions for global data (user, protection level 3)</td>
</tr>
<tr>
<td>_N_GUD4_DEF</td>
<td>Freely definable</td>
</tr>
<tr>
<td>_N_GUD5_DEF</td>
<td>Contains definitions for measuring cycles (Siemens, protection level 0)</td>
</tr>
<tr>
<td>_N_GUD6_DEF</td>
<td>Contains definitions for measuring cycles (Siemens, protection level 0)</td>
</tr>
<tr>
<td>_N_GUD7_DEF</td>
<td>Contains definitions for standard cycles (Siemens, protection level 0) or freely definable without standard cycles</td>
</tr>
<tr>
<td>_N_GUD8_DEF</td>
<td>Freely definable</td>
</tr>
<tr>
<td>_N_GUD9_DEF</td>
<td>Freely definable</td>
</tr>
</tbody>
</table>
Defining user data (GUD)

1. Save module _N_INITIAL_INI.
2. Creating a definition file for user data
   - on an external PC (SW 4 and lower)
   - in the Services operating area (SW 5 and higher)

Predefined file names are provided (see previous page):

   _N_SGUD_DEF
   _N_MGUD_DEF
   _N_UGUD_DEF
   _N_GUD4_DEF ... _N_GUD9_DEF

Files with these names can contain definitions for GUD variables.

Programming

The GUD variables are programmed with the DEF command:

```
DEF scope preproc. stop type name[.., ...]=value
```

Explanation

<table>
<thead>
<tr>
<th>Scope</th>
<th>Range identifies the variable as a GUD variable and defines its validity scope:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCK</td>
<td>Throughout NCK</td>
</tr>
<tr>
<td>CHAN</td>
<td>Throughout channel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preproc. stop</th>
<th>Optional attribute preprocessing stop:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNR</td>
<td>Preprocessing stop while reading</td>
</tr>
<tr>
<td>SYNW</td>
<td>Preprocessing stop while writing</td>
</tr>
<tr>
<td>SYNRW</td>
<td>Preprocessing stop while reading/writing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td></td>
</tr>
<tr>
<td>AXIS</td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
</tr>
</tbody>
</table>
3.4 Defining user data

<table>
<thead>
<tr>
<th>Name</th>
<th>Variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[..., ...]</td>
<td>Optional run limits for array variables</td>
</tr>
<tr>
<td>Value</td>
<td>Optional preset value, two or more values for arrays, separated by commas</td>
</tr>
<tr>
<td></td>
<td>REF (w1), SET(w1, W2, ...), (w1, w2, ...)</td>
</tr>
<tr>
<td></td>
<td>Initialization values are not possible for type Frame</td>
</tr>
</tbody>
</table>

3. Load the definition file in the program memory of the control.

The control always creates a default directory _N_DEF_DIR. This name is entered as the path in the header of the GUD definition file and evaluated when read in via the RS-232 interface.

Programming example

Example of a definition file, global data (Siemens):

```plaintext
%_N_SGUD_DEF

; $PATH=/_N_DEF_DIR

DEF NCK REAL RTP ; Retraction plane
DEF CHAN INT SDIS ; Safety clearance
M30
```
4. Activating definition files

- **SW 4 and lower**
  
  *Before read-in of the _N_INITIAL_INI, save all programs, frames, and machine data because the static memory will be formatted*

  The definition file is only reactivated on read-in of the _N_INITIAL_INI file.

- **SW 5 and higher**
  
  When the GUD definition file is loaded into the NC (“Load” soft key), it becomes active. See “Automatic activation ...”

5. Data storage

When the file _N_COMPLETE_GUD is archived from the working memory, only the data contained in the file are saved. The definition files created for the global user variables must be archived separately.

The variable assignments to global user data are also stored in _N_INITIAL_INI, the names must be identical with the names in the definition files.

**Example of a definition file for global data**

(machine manufacturer):

```plaintext
%_N_MGUD_DEF
;$PATH=/_N_DEF_DIR
;Global data definitions of the machine manufacturer
DEF NCK SYNRW INT QUANTITY ;Implicit preprocessing stop during read/write
 ;Spec. data available in the control
 ;Access from all channels
DEF CHAN INT TOOLTABLE[100] ;Tool table for channel-spec. image
 ;of the tool number at magazine locations
M30 ;Separate table created for each channel
```
3.5 Defining protection levels for user data (GUD)

Programming

Protection levels for the whole module are specified in the headers

\%
\_N\_MGUD\_DEF ; Module type

;\$PATH=/\_N\_DEF\_DIR ; Path

APR n APW n ; Protection levels on a separate line

Explanation

<table>
<thead>
<tr>
<th>APW n</th>
<th>Write access protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR n</td>
<td>Read access protection</td>
</tr>
<tr>
<td>n</td>
<td>Protection level n</td>
</tr>
<tr>
<td></td>
<td>from 0 or 10 (highest level) to 7 or 17 (lowest level)</td>
</tr>
</tbody>
</table>

Meaning of the protection levels:

<table>
<thead>
<tr>
<th>0</th>
<th>or</th>
<th>10</th>
<th>SIEMENS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>or</td>
<td>11</td>
<td>OEM_HIGH</td>
</tr>
<tr>
<td>2</td>
<td>or</td>
<td>12</td>
<td>OEM_LOW</td>
</tr>
<tr>
<td>3</td>
<td>or</td>
<td>13</td>
<td>Final user</td>
</tr>
<tr>
<td>4</td>
<td>or</td>
<td>14</td>
<td>Key switch 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>or</td>
<td>17</td>
<td>Keystitch 0</td>
</tr>
</tbody>
</table>

APW 0-7, APR 0-7

The module variables cannot be written/read via the NC program or in MDA mode.

APW 10-17, APR 10-17:

The module variables can be written/read via the NC program or in MDA mode.

Note

To protect a complete file, the commands must be placed before the first definitions in the file. In other cases, they go into the REDEF instruction of the relevant data.
Function
Access criteria can be defined for GUD modules to protect them against manipulation. In cycles GUD variables can be queried that are protected in this way from change via the HMI or from the program. The access protection applies to all variables defined in this module.
When an attempt is made to access protected data, the control outputs an appropriate alarm.

When a GUD definition file is first activated any defined access authorization contained therein is evaluated and automatically re-transferred to the read/write access of the GUD definition file.

Note
Access authorization entries in the GUD definition file can restrict but not extend the required access authorization for the GUD definition file.

Example
The definition file _N_GUD7_DEF contains: APW2

a) The file _N_GUD7_DEF has value 3 as write protection. The value 3 is then overwritten with value 2.
b) The file _N_GUD7_DEF has value 0 as write protection. There is no change to it.

With the APW instruction a retrospective change is made to the file's write access.
With the APR instruction a retrospective change is made to the file's read access.

Note
If you erroneously enter in the GUD definition file a higher access level than your authorization allows, the archive file must be reimported.
3.6 Automatic activation of GUDs and MACs (SW 4.4 and higher)

Function

The definition files for GUD and macro definitions are edited
- in the Services operating area for the MMC 102/103.

If a definition file is edited in the NC, when exiting the Editor you are prompted whether the definitions are to be set active.

Sequence

The access protection level is programmed with the desired protection level in the relevant module before any variable is defined. Vocabulary words must be programmed in a separate block.

Example of a definition file with access protection write (machine manufacturer), read (keyswitch 2):

```
%_N_GUD6_DEF
;$_PATH=/_N_DEF_DIR
APR 15 APW 12 ; Protection levels for all following variables
DEF CHAN REAL_CORRVAL
DEF NCK INT MYCOUNT
...
M30 ;
```

Programming example
Example:
"Do you want to activate the definitions from file GUD7.DEF?"
"OK" -> A request is displayed asking you whether you want to restore the currently active data.
"Do you want to save the previous data in the definitions?"
"OK" -> The GUD blocks of the definition file to be processed are saved while the new definitions are activated and the restored data are loaded again.
"Abort" -> The new definitions are activated while the old data are lost.
"Abort" -> The changes made in the definition file are canceled and the associated data block is not changed.

Unload
If a definition file is unloaded, the associated data block is deleted after a query is displayed.

Load
If a definition file is loaded, a prompt is displayed asking whether to activate the file or retain the data. If you do not activate, the file is not loaded.

If the cursor is positioned on a loaded definition file, the soft key labeling changes from "Load" to "Activate" to activate the definitions. If you select "Activate", another prompt is displayed asking whether you want to retain the data.

Data is only saved for variable definition files, not for macros.
3.7 Data-specific protection level change for machine and setting data

3.7.1 Change

Programming

REDEF Machine data/setting data protection level

Explanation

<table>
<thead>
<tr>
<th>Protection level:</th>
<th>Write access protection</th>
<th>Read access protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>APW n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APR n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function

The user change the protection levels. Only lower priority protection levels can be assigned in the machine data, and higher priority protection levels in the setting data.

The passwords are required for redefinition by the user.

Additional notes (MMC 103)

If there is not enough memory capacity for activating the definition file, once the memory size has been changed, the file must be transferred from the NC to the MMC and back into the NC again to activate it.
3.7.2 Undoing a change

To undo a change to the protection levels, the original protection levels must be written back again.

Programming example

Changing rights in individual MDs

```plaintext
%_N_SGUD_DEF
;PATH=/_N_DEF_DIR
REDEF $MA_CTRLOUT_SEGMENT_NR APR 2 APW 2
REDEF $MA_ENC_SEGMENT_NR APR 2 APW 2
REDEF $SN_JOG_CONT_MODE_LEVELTRIGGRD APR 2 APW 2
M30
```

3.8 Changing attributes of NC language elements (SW 6.4 and higher)

The REDEF instruction available as from SW 6.4 makes the functions described in the previous subsections for defining data objects and protection levels into a general interface for setting attributes and values.
### Programming

**REDEF NC language element attribute value**

### Explanation

<table>
<thead>
<tr>
<th>NC language element</th>
<th>This includes:</th>
<th>Attribute</th>
<th>Permissible for:</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUD</td>
<td>R parameters, Machine data/setting data, Synchronized variables ($AC_PARAM, $AC_MARKER, $AC_TIMER)</td>
<td>INIPO</td>
<td>GUD, R parameters, synchronized variables</td>
</tr>
<tr>
<td></td>
<td>System variables that can be written from part progs. (see Appendix)</td>
<td>INIRE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User frames (G500, etc.)</td>
<td>INICF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Magazine/tool configurations</td>
<td>PRLOC</td>
<td>Setting data</td>
</tr>
<tr>
<td></td>
<td>GCode, predefined functions, macros</td>
<td>SYNR</td>
<td>GUD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYNW</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYNRW</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>APW</td>
<td>Machine and setting data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APR</td>
<td></td>
</tr>
</tbody>
</table>
3.8 Changing attributes of NC language elements (SW 6.4 and higher)

<table>
<thead>
<tr>
<th>Value</th>
<th>Optional parameters for attributes INIPO, INIRE, INICF, PRLOC: Subsequent start values</th>
</tr>
</thead>
</table>

**Forms:**
- **Single values**
  - e.g. 5
- **Value list**
  - e.g. (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) for variable with 10 elements
- **REP (w1)**
  - with w1: value list to be repeated for variable with two or more elements, e.g. REP(12)
- **SET (w1, w2, w3, ...)**
  - or
  - (w1, w2, w3, ...) value list

**n** Required parameter protection level for attributes for APR or APW

For **GUD**, the definition can contain a start value (DEF NCK INT _MYGUD=5). If this start value is not stated (e.g. in DEF NCK INT _MYINT), the start value can be defined subsequently in the REDEF instruction.

Cannot be used for R parameters and system parameters.

Only constants can be assigned. Expressions are not permitted values.
3.8 Changing attributes of NC language elements (SW 6.4 and higher)

Meanings of the attributes

**INIPO**

Init for Power ON
The data are overwritten with the default(s) on battery-back restart of the NC.

**INIRE**

Init for operator panel front reset or TP end
At the end of a main program, for example, with M2, M30, etc. or on cancellation with the operator panel front reset, the data are overwritten with the defaults. INIRE also applies for INIPO.

**INICF**

Init on NewConf request or TP command NEWCONF
On NewConf request or TP command NEWCONF, the data are overwritten with the default values. INICF also applies to INIRE and INIPO.

The user is responsible for synchronization of the events triggering initialization. For example, if an end of parts program is executed in two different channels, the variables are initialized in each. That affects global and axial data!
3.8 Changing attributes of NC language elements (SW 6.4 and higher)

PRLOC

Only program-local change
If the data is changed in a parts program, subprogram, cycle, or ASUB, it will be restored to its original value at the end of the main program (end with, for example, M2, M30, etc. or on cancellation by operator panel front reset).
This attribute is only permissible for programmable setting data.

SYNR
SYNW
SYNRW

Only possible for GUD:
Preprocess stop while reading
Preprocess stop during write
Preprocess stop during read and write

APW
APR

Access right during write
Access right during read
For machine and setting data you can overwrite the preset access authorization subsequently. The permissible values range from
'0' (Siemens password) to
'7' (keyswitch position 0)

Supplementary conditions
The change to the attributes of NC objects can only be made after definition of the object. In particular, it is necessary to pay attention to the DEF.../REDEF sequence for GUD. (Setting data/system variables are implicitly created before the definition files are processed).
The symbol must always be defined first (implicitly by the system or by the DEF instruction) and only then can the REDEF be changed.
If two or more concurrent attribute changes are programmed, the last change is always active.
Attributes of arrays cannot be set for individual elements but only ever for the entire array:
### 3.8 Changing attributes of NC language elements (SW 6.4 and higher)

**Initialization of GUD arrays themselves is not affected.**

```
DEF NCK INT _MYGUS[10,10]
REDEF _MYGUD INIRE // ok
REDEF _MYGUD[1,1] INIRE // not possible, alarm output
    // (array value)
```

Make sure that a sufficiently large **memory for init values** (MD 18150: MM_GUD_VAL_MEM) is available when setting INI attributes for these variables. In MD 11270:

**DEFAULT_VALUES_MEM_MASK, Bit1 = 1 must be set (memory for initialization values active).**

For R and system parameters it is not possible to specify a default that deviates from the compiled value. However, resetting to the compiled value is possible with INIPO, INIRE, or INICF.

For data type FRAME of GUD it is not possible to specify a default deviating from the compiled value either (like for definition of the data item).
Programming example 1

Reset behavior with GUD

```
/ /N_DEF_DIR/ /N/lgUD_DEF
DEF NCK INT _/MYGUD1 ; Definitions
DEF NCK INT _/MYGUD2 = 2
DEF NCK INT _/MYGUD3 = 3
```

Initialization on operator panel front reset/end of parts program:

```
REDEF _/MYGUD2 INIRE ; Initialization
M17
```

This sets "_MYGUD2" back to "2" on operator panel front reset / end of parts program whereas "_MYGUD1" and "_MYGUD3" retain their value.

Programming example 2

Modal speed limitation in the parts program (setting data)

```
/ /N_DEF_DIR/ /N/lgUD_DEF
REDEF $SA_SPIND_MAX_VELO_LIMS PRLOC ; Setting data for limit speed
M17
```

```
/ /N_MPF_DIR/ /N/lg_MPF
N10 SETMS (3)
N20 G96 S100 LIMS=2500
... M30
```

Let the limit speed defined in setting data ($SA_SPIND_MAX_VELO_LIMS) speed limitation be 1200rpm. Because a higher speed can be permitted in a set-up and completely tested parts program, LIMS=2500 is programmed here. After the end of the program, the value configured in the setting data takes effect here again.
### Programmable setting data

The following SD can be initialized with the REDEF instruction:

<table>
<thead>
<tr>
<th>Number</th>
<th>Identifier</th>
<th>GCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>42000</td>
<td>$SC_THREAD_START_ANGLE</td>
<td>SF</td>
</tr>
<tr>
<td>42010</td>
<td>$SC_THREAD_RAMP_DISP</td>
<td>DITS/DITE</td>
</tr>
<tr>
<td>42400</td>
<td>$SC_PUNCH_DWELLTIME</td>
<td>PDELAYON</td>
</tr>
<tr>
<td>42800</td>
<td>$SC_SPIND.Assign_TAB</td>
<td>SETMS</td>
</tr>
<tr>
<td>43210</td>
<td>$SA_SPIND_MIN_VELO_G25</td>
<td>G25</td>
</tr>
<tr>
<td>43220</td>
<td>$SA_SPIND_MAX_VELO_G26</td>
<td>G26</td>
</tr>
<tr>
<td>43230</td>
<td>$SA_SPIND_MAX_VELO_LIMS</td>
<td>LIMS</td>
</tr>
<tr>
<td>43300</td>
<td>$SA_Assign_FEED_PER_REV_SOURCE</td>
<td>FPRAON</td>
</tr>
<tr>
<td>43420</td>
<td>$SA_WORKAREA_LIMIT_PLUS</td>
<td>G26</td>
</tr>
<tr>
<td>43430</td>
<td>$SA_WORKAREA_LIMIT_MINUS</td>
<td>G25</td>
</tr>
<tr>
<td>43510</td>
<td>$SA_FIXED_STOP_TORQUE</td>
<td>FXST</td>
</tr>
<tr>
<td>43520</td>
<td>$SA_FIXED_STOP_WINDOW</td>
<td>FXSW</td>
</tr>
<tr>
<td>43700</td>
<td>$SA_OSCILL.REVERSE_POS1</td>
<td>OSP1</td>
</tr>
<tr>
<td>43710</td>
<td>$SA_OSCILL.REVERSE_POS2</td>
<td>OSP2</td>
</tr>
<tr>
<td>43720</td>
<td>$SA_OSCILL.DWELL.Time1</td>
<td>OST1</td>
</tr>
<tr>
<td>43730</td>
<td>$SA_OSCILL.DWELL.Time2</td>
<td>OST2</td>
</tr>
<tr>
<td>43740</td>
<td>$SA_OSCILL_VELO</td>
<td>FA</td>
</tr>
<tr>
<td>43750</td>
<td>$SA_OSCILL.NUM.SPARK.CYCLES</td>
<td>OSNSC</td>
</tr>
<tr>
<td>43760</td>
<td>$SA_OSCILL.END.POS</td>
<td>OSE</td>
</tr>
<tr>
<td>43770</td>
<td>$SA_OSCILL_CTRL.MASK</td>
<td>OSCTRL</td>
</tr>
<tr>
<td>43780</td>
<td>$SA_OSCILL_IS_ACTIVE</td>
<td>OS</td>
</tr>
</tbody>
</table>

### System variables that can be written from the parts program:

Section 15.2 of this description lists the system variables. All system variables that are marked W (write) or WS (write with preprocess stop) in column parts program can be initialized with the RESET instruction.
3.9 Structuring instruction SEFORM in the Step editor
(SW 6.4 and higher)

Programming

SEFORM(STRING[128] section_name, INT level, STRING[128] icon)

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEFORM</td>
<td>Function call of structuring instruction with parameters: section_name, level, and icon</td>
</tr>
<tr>
<td>section_name</td>
<td>Identifier of the operation</td>
</tr>
<tr>
<td>level</td>
<td>Index for the main or sublevel.</td>
</tr>
<tr>
<td></td>
<td>=0 means main level</td>
</tr>
<tr>
<td></td>
<td>=1, ... means sublevel 1 to n</td>
</tr>
<tr>
<td>icon</td>
<td>Name of the icon displayed for this section.</td>
</tr>
</tbody>
</table>

Function

The SEFORM instruction is evaluated in the Step editor to generate the step view for HMI Advanced. The step view is available as from SW 6.3 on HMI Advanced and makes for better readability of the NC subprogram. The SEFORM structuring instruction supports Step editor (editor-based program support) over the three specified parameters.
Additional notes

- The SEFORM instructions are generated in the Step editor.
- The string passed with the `<section_name>` parameter is stored main-run-synchronously in the OPI variable in a similar way to the MSG instruction. The information remains until overwritten by the next SEFORM instruction. Reset and end of parts program clear the content.
- The level and icon parameters are checked by the parts program processing of the NCK but not further processed.

For more information about editor-based programming support, see:
/BAD/ Operator's Guide HMI Advanced
Protection Zones

4.1 Defining the protection zones CPROTDEF, NPROTDEF ........................................ 4-176
4.2 Activating/deactivating protection zones: CPROT, NPROT ........................................ 4-180
4.1 Defining the protection zones CPROTDEF, NPROTDEF

**Programming**

```
DEF INT NOT_USED
CPROTDEF(n, t, applim, applus, appminus)
NPROTDEF(n, t, applim, applus, appminus)
EXECUTE (NOT_USED)
```

**Explanation of the commands**

- **DEF INT NOT_USED**: Define local variable, data type integer (see Chapter 10)
- **CPROTDEF**: Channel-specific protection zones (for NCU 572/573 only)
- **NPROTDEF**: Machine-specific protection zones
- **EXECUTE**: End definition

**Explanation of the parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>Number of defined protection zone</td>
</tr>
</tbody>
</table>
| t         | TRUE = **Tool**-oriented protection zone  
            FALSE = **Workpiece**-oriented protection zone |
| applim    | Type of limit in the 3rd dimension  
            0 = No limit  
            1 = Limit in positive direction  
            2 = Limit in negative direction  
            3 = Limit in positive and negative direction |
| applus    | Value of the limit in the positive direction in the 3rd dimension |
| appminus  | Value of the limit in the negative direction in the 3rd dimension |
| NOT_USED  | Error variable has no effect in protection zones with EXECUTE |
### Function

You can use protection zones to protect various elements on the machine, their components and the workpiece against incorrect movements.

**Tool-oriented** protection zones:
For parts which belong to the tool (e.g. tool, tool carrier).

**Workpiece-oriented** protection zones:
For parts which belong to the workpiece (e.g. parts of the workpiece, clamping table, clamp, spindle chuck, tailstock).

### Sequence

Defining protection zones

Definition of the protection zones includes the following:

- CPROTDEF for channel-specific protection zones
- N PROTDEF for machine-specific protection zones
- Contour description for protection zone
- Termination of the definition with EXECUTE

You can specify a relative offset for the reference point of the protection zone when the protection zone is activated in the NC parts program.
### Reference point for contour description

The workpiece-oriented protection zones are defined in the basic coordinate system. The tool-oriented protection zones are defined with reference to the tool carrier reference point F.

### Contour definition of protection zones

The contour of the protection zones is specified with up to 11 traversing movements in the selected plane. The first traversing movement is the movement to the contour. The area to the left of the contour qualifies as the protection zone. The travel motions programmed between CPROTDEF or NPROTDEF and EXECUTE are not executed, but merely define the protection zone.

### Working plane

The required plane is selected before CPROTDEF and NPROTDEF with G17, G18, G19 and must not be altered before EXECUTE. The applicate must not be programmed between CPROTDEF or NPROTDEF and EXECUTE.

### Contour elements

The following are permitted:
- G0, G1 for straight contour elements
- G2 for clockwise circle segments (only for tool-oriented protection zones)
- G3 for counterclockwise circle segments

A maximum of four contour elements are available for defining one protection zone (max. of four protection zones) with the SINUMERIK FM-NC. With the 810D, a maximum of 4 contour elements are available for defining one protection zone (max. of four channel-specific and 4 NCK-specific protection zones).
4 11.02 Protection Zones

4.1 Defining the protection zones CPROTDEF, NPROTDEF

If a full circle describes the protection zone, it must be divided into two half circles. The order G2, G3 or G3, G2 is not permitted. A short G1 block must be inserted, if necessary.

The last point in the contour description must coincide with the first.

External protection zones (only possible for workpiece-oriented protection zones) should be defined **in the clockwise direction**.

For **dynamically balanced** protection zones (e.g. spindle chucks) you must describe the **complete contour** (and not only up to the center of rotation!).

Tool-oriented protection zones must always be **convex**. If a concave protected zone is desired, this should be subdivided into several convex protection zones.

The following must not be active while the protection zones are defined:

- Cutter radius or tool nose radius compensation,
- Transformation,
- Frame.

Nor must reference point approach (G74), fixed point approach (G75), block search stop or program end be programmed.
4.2 Activating/deactivating protection zones: CPROT, NPROT

Programming

CPROT \((n, \text{state}, x\text{Mov}, y\text{Mov}, z\text{Mov})\)

NPROT \((n, \text{state}, x\text{Mov}, y\text{Mov}, z\text{Mov})\)

Explanation of the commands and parameters

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPROT</td>
<td>Call channel-specific protection zone (for NCU 572/573 only)</td>
</tr>
<tr>
<td>NPROT</td>
<td>Call machine-specific protection zone</td>
</tr>
<tr>
<td>n</td>
<td>Number of protection zone</td>
</tr>
<tr>
<td>state</td>
<td>Status parameter</td>
</tr>
<tr>
<td></td>
<td>0 = Deactivate protection zone</td>
</tr>
<tr>
<td></td>
<td>1 = Preactivate protection zone</td>
</tr>
<tr>
<td></td>
<td>2 = Activate protection zone</td>
</tr>
<tr>
<td>x\text{Mov}, y\text{Mov}, z\text{Mov}</td>
<td>Move defined protection zone on the geometry axes</td>
</tr>
</tbody>
</table>

Function

Activating and preactivating previously defined protection zones for collision monitoring and deactivating protection zones.

The maximum number of protection zones which can be active simultaneously on the same channel is defined in machine data.

If no tool-oriented protection zone is active, the tool path is checked against the workpiece-oriented protection zones.

If no workpiece-oriented protection zone is active, protection zone monitoring does not take place.
4.2 Activating/deactivating protection zones: CPROT, N PROT

Sequence

Activation status
A protection zone is generally activated in the parts program with status = 2.

The status is always channel-specific even for machine-oriented protection zones.

If a PLC user program provides for a protection zone to be effectively set by a PLC user program, the required preactivation is implemented with status = 1.

The protection zones are deactivated and therefore disabled with Status = 0. No offset is necessary.

Offset of protection zones on (pre)activation
The offset can take place in 1, 2, or 3 dimensions. The offset refers to:
• the machine zero in workpiece-specific protection zones,
• the tool carrier reference point F in tool-specific protection zones.

Additional notes
Protection zones can be activated straight after booting and subsequent reference point approach. The system variable $SN_PA_ACTIV_IMMED [n] or $SN_PA_ACTIV_IMMED[n] = TRUE must be set for this. They are always activated with Status = 2 and have no offset.
Multiple activation of protection zones

A protection zone can be active simultaneously in several channels (e.g. tailstock where there are two opposite sides).
The protection zones are only monitored if all geometry axes have been referenced. The following rules apply:

- The protection zone cannot be activated simultaneously with different offsets in a single channel.
- Machine-oriented protection zones must have the same orientation on both channels.

Programming example

Possible collision of a milling cutter with the measuring probe is to be monitored on a milling machine. The position of the measuring probe is to be defined by an offset when the function is activated.
The following protection zones are defined for this:

- A machine-specific and a workpiece-oriented protection zone for both the measuring probe holder (n-SB1) and the measuring probe itself (n-SB2).
- A channel-specific and a tool-oriented protection zone for the milling cutter holder (c-SB1), the cutter shank (c-SB2) and the milling cutter itself (c-SB3).

The orientation of all protection zones is in the Z direction.

The position of the reference point of the measuring probe on activation of the function must be X = –120, Y = 60 and Z = 80.
**11.02 Protection Zones**

### 4.2 Activating/deactivating protection zones: CPROT, NPROT

#### Definition of protection zones

**Definition of an auxiliary variable**

- **G17**

- **NPROTDEF(1,FALSE,3,10,–10)**
  
  Protection zone n–SB1

  ```plaintext
  G01 X0 Y–10
  X40
  Y10
  X0
  Y–10
  EXECUTE(PROTECTB)
  ```

- **NPROTDEF(2,FALSE,3,5,–5)**
  
  Protection zone n–SB2

  ```plaintext
  G01 X40 Y–5
  X70
  Y5
  X40
  Y–5
  EXECUTE(PROTECTB)
  ```

- **CPROTDEF(1,TRUE,3,0,–100)**
  
  Protection zone c–SB1

  ```plaintext
  G01 X–20 Y–20
  X20
  Y20
  X–20
  Y–20
  EXECUTE(PROTECTB)
  ```

- **CPROTDEF(2,TRUE,3,–100,–150)**
  
  Protection zone c–SB2

  ```plaintext
  G01 X0 Y–10
  G03 X0 Y10 J10
  X0 Y–10 J–10
  EXECUTE(PROTECTB)
  ```

- **CPROTDEF(3,TRUE,3,–150,–170)**
  
  Protection zone c–SB3

  ```plaintext
  G01 X0 Y–27,5
  G03 X0 Y27,5 J27,5
  X0 Y27,5 J–27,5
  EXECUTE(PROTECTB)
  ```

#### Activation of protection zones:

- **NPROT(1,2,–120,60,80)**
  
  Activate protection zone n–SB1 with offset

- **NPROT(2,2,–120,60,80)**
  
  Activate protection zone n–SB2 with offset

- **CPROT(1,2,0,0,0)**
  
  Activate protection zone c–SB1 with offset

- **CPROT(2,2,0,0,0)**
  
  Activate protection zone c–SB2 with offset

- **CPROT(3,2,0,0,0)**
  
  Activate protection zone c–SB3 with offset
4.2 Activating/deactivating protection zones: CPROT, NPROT

Notes
## Special Motion Commands

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Approaching coded positions, CAC, CIC, CDC, CACP, CACN</td>
<td>5-186</td>
</tr>
<tr>
<td>5.2</td>
<td>Spline interpolation</td>
<td>5-187</td>
</tr>
<tr>
<td>5.3</td>
<td>Compressor COMPON/COMPCURV/COMPCAD (SW 6.2)</td>
<td>5-196</td>
</tr>
<tr>
<td>5.4</td>
<td>Polynomial interpolation – POLY, POLYPATH (SW 5 and higher)</td>
<td>5-204</td>
</tr>
<tr>
<td>5.5</td>
<td>Settable path reference, SPATH, UPATH (SW 4.3 and higher)</td>
<td>5-211</td>
</tr>
<tr>
<td>5.6</td>
<td>Measurements with touch trigger probe, MEAS, MEAW</td>
<td>5-215</td>
</tr>
<tr>
<td>5.7</td>
<td>Extended measuring function MEASA, MEAWA, MEAC (SW 4 and higher, option)</td>
<td>5-218</td>
</tr>
<tr>
<td>5.8</td>
<td>Special functions for OEM users</td>
<td>5-228</td>
</tr>
<tr>
<td>5.9</td>
<td>Programmable motion end criterion (SW 5.1 and higher)</td>
<td>5-229</td>
</tr>
<tr>
<td>5.10</td>
<td>Programmable servo parameter block (SW 5.1 and higher)</td>
<td>5-232</td>
</tr>
</tbody>
</table>
5.1 Approaching coded positions, CAC, CIC, CDC, CACP, CACN

Explanation of the commands

- **CAC(n)**: Approach coded positions absolutely
- **CIC(n)**: Approach coded position incrementally by n spaces in plus direction (+) or in minus direction (−)
- **CDC(n)**: Approach coded position via shortest possible route (rotary axes only)
- **CACP(n)**: Approach coded position absolutely in positive direction (rotary axes only)
- **CACN(n)**: Approach coded position absolutely in negative direction (rotary axes only)

(n): Position numbers 1, 2, ... max. 60 positions for each axis

Sequence

You can enter a maximum of 60 (0 to 59) positions in special position tables for two axes in machine data.

For an example of a typical position table see diagram.

Further details

If an axis is situated between two positions, it does not traverse in response to an incremental position command with CIC (...). It is always advisable to program the first travel command with an absolute position value.

Programming example

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>FA[B]=300</td>
<td>Feed for positioning axis B</td>
</tr>
<tr>
<td>N20</td>
<td>POS[B]=CAC(10)</td>
<td>Approach coded position 10 (absolutely)</td>
</tr>
<tr>
<td>N30</td>
<td>POS[B]=CIC(-4)</td>
<td>Travel 4 spaces back from the current position</td>
</tr>
</tbody>
</table>

Table 1 (linear axis)

<table>
<thead>
<tr>
<th>Position number</th>
<th>Position value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>27.3</td>
</tr>
<tr>
<td>2</td>
<td>40.7</td>
</tr>
<tr>
<td>3</td>
<td>112</td>
</tr>
<tr>
<td>4</td>
<td>112 mm</td>
</tr>
</tbody>
</table>

Indexing axis:

<table>
<thead>
<tr>
<th>Position number</th>
<th>Position value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>315 deg</td>
</tr>
</tbody>
</table>

Table 1 (rotary axis)
5.2 Spline interpolation

**Introduction**

The spline interpolation function can be used to link series of points along smooth curves. Splines can be applied, for example, to create curves using a sequence of digitized points.

There are several types of spline with different characteristics, each producing different interpolation effects. In addition to selecting the spline type, the user can also manipulate a range of different parameters. Several attempts are normally required to obtain the desired pattern.

**Programming**

ASPLINEX Y Z A B C  
or  
BSPLINE X Y Z A B C  
or  
CSPLINE X Y Z A B C

**Function**

In programming a spline, you link a series of points along a curve.

You can select one of three spline types:
- A spline (akima spline)
- B spline (non-uniform, rational basis spline, NURBS)
- C spline (cubic spline)
5.2 Spline interpolation

**Additional notes**

A, B and C splines are modally active and belong to the group of motion commands. The tool radius offset may be used. Collision monitoring is carried out in the projection in the plane.

Axes that are to interpolate in the spline grouping are selected with command SPLINEPATH (further details on the following pages).

**Sequence**

**A SPLINE**

The A spline (Akima spline) passes exactly through the intermediate points. While it produces virtually no undesirable oscillations, it does not create a continuous curve in the interpolation points.

The akima spline is local, i.e. a change to an interpolation point affects only up to six adjacent points.

The primary application for this spline type is therefore the interpolation of digitized points. Supplementary conditions can be programmed for akima splines (see below for more information). A polynomial of third degree is used for interpolation.
### B SPLINE

With a B spline, the programmed positions are not intermediate points, but merely check points of the spline, i.e. the curve is "drawn towards" the points, but does not pass directly through them.

The lines linking the points form the check polygon of the spline. B splines are the optimum means for defining tool paths on sculptured surfaces. Their primary purpose is to act as the interface to CAD systems. A third degree B spline does not produce any oscillations in spite of its continuously curved transitions.

Programmed supplementary conditions (please see below for more information) have no effect on B splines. The B spline is always tangential to the check polygon at its start and end points.

**Point weight:**
A weight can be programmed for every interpolation point.

**Programming:**
\[ PW = n \]

**Value range:**
0 <= n <= 3; in steps of 0.0001

**Effect:**
- n > 1  The check point exerts more "force" on the curve
- n < 1  The check point exerts less "force" on the curve

**Spline degree:**
A third degree polygon is used as standard, but a second degree polygon is also possible.

**Programming:**
\[ SD = 2 \]
Distance between nodes:
Node distances are appropriately calculated internally in the control, but the system is also capable of processing user-programmed node distances.

Programming:
\[ \text{PL} = \text{Value range as for path dimension} \]

Example of B spline:

<table>
<thead>
<tr>
<th>All weights 1</th>
<th>Different weights</th>
<th>Check polygon</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10 G1 X0 Y0 F300 G64</td>
<td>N10 G1 X0 Y0 F300 G64</td>
<td>N10 G1 X0 Y0 F300 G64</td>
</tr>
<tr>
<td>N20 BSPLINE</td>
<td>N20 BSPLINE</td>
<td>N20 BSPLINE</td>
</tr>
<tr>
<td>N30 X10 Y20</td>
<td>N30 X10 Y20 PW=2</td>
<td>N30 X10 Y20</td>
</tr>
<tr>
<td>N40 X20 Y40</td>
<td>N40 X20 Y40</td>
<td>N40 X20 Y40</td>
</tr>
<tr>
<td>N50 X30 Y30</td>
<td>N50 X30 Y30 PW=0.5</td>
<td>N50 X30 Y30</td>
</tr>
<tr>
<td>N60 X40 Y45</td>
<td>N60 X40 Y45</td>
<td>N60 X40 Y45</td>
</tr>
<tr>
<td>N70 X50 Y0</td>
<td>N70 X50 Y0</td>
<td>N70 X50 Y0</td>
</tr>
</tbody>
</table>

C SPLINE

In contrast to the akima spine, the cubic spline is continuously curved in the intermediate points. It tends to have unexpected fluctuations however. It can be used in cases where the interpolation points lie along an analytically calculated curve. C splines use third degree polynomials.

The spline is not local, i.e. changes to an interpolation point can influence a large number of blocks (with gradually decreasing effect).
Supplementary conditions

The following supplementary conditions apply only to akima and cubic splines (A and C splines).

The transitional response (start and end) of these spline curves can be set via two groups of instructions consisting of three commands each.

Explanation of the commands

Start of spline curve:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAUTO</td>
<td>No command input; start is determined by the position of the first point</td>
</tr>
<tr>
<td>BNAT</td>
<td>Zero curvature</td>
</tr>
<tr>
<td>BTAN</td>
<td>Tangential transition to preceding block (initial setting)</td>
</tr>
</tbody>
</table>

End of spline curve:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAUTO</td>
<td>No command input; end is determined by the position of the last point</td>
</tr>
<tr>
<td>ENAT</td>
<td>Zero curvature</td>
</tr>
<tr>
<td>ETAN</td>
<td>Tangential transition to next block (initial setting)</td>
</tr>
</tbody>
</table>
Example

C spline, zero curvature at start and end

N10 G1 X0 Y0 F300
N15 X10
N20 BNAT ENAT

C spline, at start and end
Zero curvature
N30 CSPLINE X20 Y10
N40 X30
N50 X40 Y5
N60 X50 Y15
N70 X55 Y7
N80 X60 Y20
N90 X65 Y20
N100 X70 Y0
N110 X80 Y10
N120 X90 Y0
N130 M30
What does which spline do?

Comparison of three spline types with identical interpolation points:

A spline (akima spline)
B spline (Bezier spline)
C spline (cubic spline)

Spline grouping

Up to eight path axes can be involved in a spline interpolation grouping. The SPLINEPATH instruction defines which axes are to be involved in the spline. The instruction is programmed in a separate block. If SPLINEPATH is not explicitly programmed, then the first three axes in the channel are traversed as the spline grouping.
5.2 Spline interpolation

**Programming**

SPLINEPATH(n,X,Y,Z,...)

**Explanation**

| SPLINEPATH(n,X,Y,Z,...) | n = 1, fixed value  
|-------------------------|---------------------  
|                       | X,Y,Z,... path axis names |

**Example**

Spline grouping with three path axes

```
N10 G1 X10 Y20 Z30 A40 B50 F350
N11 SPLINEPATH(1,X,Y,Z)  
N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60  
N14 X30 Y40 Z50 A60 B70  
...  
N100 G1 X... Y...  
```

Spline grouping  
C spline  
Interpolation points  
Deselection of spline interpolation
Settings for splines

The G codes ASPLINE, BSPLINE and CSPLINE link block endpoints with splines.

For this purpose, a series of blocks (endpoints) must be simultaneously calculated.

The buffer size for calculations is ten blocks as standard.

Not all block information is a spline endpoint. However, the control requires a certain number of spline endpoint blocks from ten blocks.

They are as follows for:

<table>
<thead>
<tr>
<th>Type</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>A spline</td>
<td>At least 4 blocks out of every 10 must be spline blocks. These do not include comment blocks and parameter calculations.</td>
</tr>
<tr>
<td>B spline</td>
<td>At least 6 blocks out of every 10 must be spline blocks. These do not include comment blocks and parameter calculations.</td>
</tr>
<tr>
<td>C spline</td>
<td>From each 10 blocks at least the contents of machine data $MC_CUBIC_SPLINE_BLOCKS+1$ must be spline blocks (also in standard case 9). The number of points must be entered in machine data $MC_CUBIC_SPLINE_BLOCKS$ (standard value 8) which are used for calculating the spline segment.</td>
</tr>
</tbody>
</table>

An alarm is output if the tolerated value is exceeded and likewise when one of the axes involved in the spline is programmed as a positioning axis.
5.3 Compressor COMPON/COMPCURV/COMPCAD (SW 6.2)

Programming

COMPON/COMPCURV/COMPCAD
COMPOF

Explanation

COMPON/COMPCURV/COMPCAD
COMPOF

Compressor ON
Compressor OFF

Function

With G code COMPON block transitions are only constant in speed, while acceleration of the participating axes can be in jumps at block transitions. This can increase oscillation on the machine.

**SW 4.4 and higher:**
With G code COMPCURV, the block transitions are with constant acceleration. This ensures both smooth velocity and acceleration of all axes at block transitions.

**SW 6.2 and higher:**
Another compression can be selected with the G code COMPCAD. Its surface finish and speed can be optimized, and the interpolation precision can be determined via machine data. COMPCAD is computation- and memory-intensive and should only be used if it was not possible to improve the surface by means of the CAD/CAM program.

Properties:
- COMPCAD generates polynomial blocks that merge into one another with constant acceleration.
- With adjacent paths, deviations head in the same direction.
- A limit angle can be defined with setting data $SC_CRIT_SPLINE_ANGLE; COMPCAD will leave the corners from this angle.
5.3 Compressor COMPON/COMPCURV/COMPCAD (SW 6.2)

- The number of blocks to be compressed is not limited to 10.
- COMPCAD eliminates poor surface transitions. In doing so, however, the tolerances are largely adhered to but the corner limit angle is ignored.
- The rounding function G642 can also be used.

**SW 6.2 and higher:**
The compressors COMPON, COMPCURV and COMPCAD are extended in a way that even NC programs for which orientation was programmed via directional vectors, can be compressed respecting a specifiable tolerance.
The function "Compressor for orientations" is only available if the orientation transformation option is available.
The restrictions mentioned above under "Conditions of usage" have been relieved to allow position values via parameter settings now also.

NC block structure in general:
N10 G1 X=<...> Y=<...> Z=<...> A=<...> B=<...> F=<...> ; comment

Active orientation transformation (TRAORI) being active, the following kinematics-independent programming options for the tool direction of 5-axis machines are possible.

1. Program the direction vector via:
   A3=< ...> B3=< ... > C3=< ... >
2. Program the Euler angle or RPY angle:
   A2=< ...> B2=< ... > C2=< ... >

The orientation motion is only compressed when the large circle interpolation is active, i.e. the tool orientation is changed in the plane which is determined by start and end orientation.
A large circle interpolation is performed under the following conditions:

1. For MD 21104: ORI_IPO_WITH_G_CODE = FALSE, if ORIWKS is active and orientation is programmed as vector (with A3, B3, C3 or A2, B2, C2).

2. For MD 21104: ORI_IPO_WITH_G_CODE = TRUE, if ORIVECT or ORIPLANE are active. Tool orientation can be programmed either as direction vector or with rotary axis positions. If one of the G–codes ORICONxx or ORICURVE is active or if polynomials are programmed for the orientation angle (PO[PHI] and PO[PSI]) a large circle interpolation is not performed, i.e., blocks of this type are not compressed.

For 6-axis machines you can program the tool rotation in addition to the tool orientation. You can program the angle of rotation with the identifier THETA (THETA=<...>).

NC blocks in which additional rotation is programmed, can only be compressed if the angle of rotation changes linear, meaning that you must not program a polynomial with PO[THT]=(...) for the angle of rotation.

NC block structure in general:

```
N... X=<...> Y=<...> Z=<...> A3=<...> B3=<...> C3=<...> THETA=<...> F=<...>
```

or

```
N... X=<...> Y=<...> Z=<...> A2=<...> B2=<...> C2=<...> THETA=<...> F=<...>
```

If tool orientation is specified via rotary axis positions, e.g. as:

```
N... X=<...> Y=<...> Z=<...> A=<...> B=<...> THETA=<...> F=<...>
```

the compression will be performed in two different ways, depending on whether a large circle interpolation is performed or not. If large circle interpolation is not performed, the compressed orientation change is represented by axial polynomials for the rotary axes.
Accuracy
You can compress NC blocks only if you allow the contour to deviate from the programmed contour.
You can set the maximal deviation as a compressor tolerance in the setting data. The higher these allowed tolerances are set, the more blocks can be compressed.

Axis precision
For each axis, the compressor creates a spline curve which deviates from the programmed end points of each axis by max. the tolerance set with the axial MD.

Contour precision
It controls the max. geometrical contour deviations (geometry axes) and the tool orientation. It is done via the setting data for:
1. Max. tolerance for the contour
2. Max. angular displacement for tool orientation
3. Max. angular displacement for the angle of tool rotation (only available for 6–axis machines)

With the channel-specific MD 20482 COMPRESSOR_MODE, you can set tolerance specifications:

0: axis precision: axial tolerances for all axes (geometry axes and orientation axes).
1: Contour precision: Specification of the contour tolerance (1.), the tolerance for orientation via axial tolerances (a.).
2: Specification of the max. angular displacement for tool orientation (2.), tolerance for the contour via axial tolerances (a.).
3: Specification of the contour tolerance with (1.) and specification of the max. angular displacement for tool orientation with (2.).

You can specify the angular displacement of the tool orientation only if an orientation transformation (TRAORI) is active.
Activation

You can activate "Compressor for orientations" via one of the following commands:

COMPON, COMPCURV (COMPCAD not possible).

References: /FB3/, F2: "3-axis to 5-axis transformation"

Machine manufacturer

Three sets of machine data are provided for the compressor function:

- $MC_COMPRESS_BLOCK_PATH_LIMIT
  A maximum path length is set. All the blocks along this path are suitable for compression. Larger blocks are not compressed.

- $MA_COMPRESS_POS_TOL
  A tolerance can be set for each axis. The generated spline curve does not deviate by more than this value from the programmed end points. The higher these values are set, the more blocks can be compressed.

- $MC_COMPRESS_VELO_TOL
  The maximum permissible path feed deviation with active compressor can be preset in conjunction with FLIN and FCUB.

Special features with COMPCAD:

- $MN_MM_EXT_PROG_BUFFER_SIZE should be large, e.g. 100 (KB).
- $MC_COMPRESS_BLOCK_PATH_LIMIT must be significantly increased in value, e.g. 50 (mm).
- $MC_MM_NUM_BLOCKS_IN_PREP must be $\geq 60$, to allow machining of much more than 10 points.
- FLIN and FCUB cannot be used.

Recommended for large block lengths and optimum velocity:

- $MC_MM_MAX_AXISPOLY_PER_BLOCK = 5$
- $MC_MM_PATH_VELO_SEGMENTS = 5$
- $MC_MM_ARCLENGTH_SEGMENTS = 10$. 
As a rule, CAD/CAM systems provide linear blocks that meet the programmed accuracy. With complex contours this leads to a considerable amount of data and to short path sections. These short path sections restrict the execution speed. With the compressor a certain number (max. 10) of these short path sections can be joined together to form one path section.

The modal G code COMPON or COMPCURV activates an "NC block compressor". With linear interpolation, this function groups a number of straight blocks (number is restricted to 10) and approaches them by means of third degree polynomials (COMPON), or fifth degree polynomials (COMPCURV), within an error tolerance range specified via machine data. In this way, the NC processes one large motion block rather than a large number of small ones.

**Conditions for usage:**
This compression operation can only be executed on linear blocks (G1). It is interrupted by any other type of NC instruction, e.g. an auxiliary function output, but not by parameter calculations. Only those blocks containing nothing more than the block number, G1, axis addresses, feed and comments are compressed. All other blocks are executed unchanged (no compression). Variables may not be used.
5
Special Motion Commands
5.3 Compressor COMPON/COMPCURV/COMPCAD (SW 6.2)

Example COMPON

```
N10 COMPON
N11 G1 X0.37 Y2.9 F600
N12 X16.87 Y-4.698
N13 X16.865 Y-4.72
N14 X16.91 Y-4.799
...
N1037 COMPOF
```

All blocks are compressed for which a simple syntax is sufficient.

E.g.
```
N19 X0.103 Y0. 20.
N20 X0.102 Y-0.018
N21 X0.097 Y-0.036
N22 X0.089 Y-0.052
N23 X0.078 Y-0.067
```

Not compressed are e.g. extended addresses such as C=100 or A=AC(100).

From NC SW 6.3: Motion blocks with extended addresses are now also compressed.

Example COMPCAD

```
G00 X30 Y6 Z40
G1 F10000 G642
SOFT
COMPCAD
STOFFIFO
N24050 Z32.499
N24051 X41.365 Z32.500
N24052 X43.115 Z32.497
N24053 X43.365 Z32.477
N24054 X43.556 Z32.449
N24055 X43.818 Z32.387
N24056 X44.076 Z32.300
...
COMPOF
G00 Z50
M30
```

Compressor OFF
### Example "Compressor for orientations"

The following program example shows how to compress a circle which is approached by a polygon definition. A synchronous tool orientation moves on the outside of a taper at the same time. Although the programmed orientation changes are executed one after the other, but in an unsteady way, the compressor generates a smooth motion of the orientation.

```
DEF INT NUMBER = 60
DEF REAL RADIUS = 20
DEF INT COUNTER
DEF REAL ANGLE
N10 G1 X0 Y0 F5000 G64
$SC_COMPRESS_CONTOUR_TOL = 0.05
$SC_COMPRESS_ORI_TOL = 5
TRAORI
COMPCURV
N100 X0 Y0 A3=0 B3=-1 C3=1
N110 FOR COUNTER = 0 TO NUMBER
N120 ANGLE = 360 * COUNTER / NUMBER
N130 X=RADIUS*COS(WINKEL) Y=RADIUS*SIN(ANGLE)
   A3=SIN(ANGLE) B3=-COS(ANGLE) C3=1
N140 ENDFOR
...
```

max. contour deviation 0.05mm
max. deviation of the orientation 5 degrees

A polygon-generated circle is traversed, while the orientation moves on a taper around the Z axis at an arc angle of 45 degrees.
5.4 Polynomial interpolation – POLY, POLYPATH (SW 5 and higher)

The control system is capable of traversing curves (paths) in which every selected path axis is operating as a function of up to SW 5 (polynomial, max. third degree), from SW 6 (polynomial, max. fifth degree).

The equation used to express the polynomial function is generally as follows:

- For SW 5 and lower:
  \[ f(p) = a_0 + a_1p + a_2p^2 + a_3p^3 \]
- For SW 6 and higher:
  \[ f(p) = a_0 + a_1p + a_2p^2 + a_3p^3 + a_4p^4 + a_5p^5 \]

The letters have the following meaning:

- \( a_n \): Constant coefficients
- \( p \): Parameters

By assigning concrete values to these coefficients, it is possible to generate a wide variety of curve shapes such as line, parabola and power functions.

By setting the coefficients as \( a_2 = a_3 = 0 \) (SW 5 and lower) or \( a_2 = a_3 = a_4 = a_5 = 0 \) (SW 6 and higher) it is possible to create, e.g. a straight line with

\[ f(p) = a_0 + a_1p \]

Meanings:

- \( a_0 \): Axis position at end of preceding block
- \( a_1 \): Difference between axis position at end of the definition range (PL) and start position

Definition

Polynomial interpolation (POLY) is not one of the real types of spline interpolation. Its main purpose is to act as an interface for programming externally generated spline curves where the spline sections can be programmed directly.

This mode of interpolation relieves the NC of the task of calculating polynomial coefficients. It can be applied optimally in cases where the coefficients are supplied directly by a CAD system or postprocessor.
Polynomial interpolation belongs to the first G group along with G0, G1, G2, G3, A spline, B spline and C spline. If it is active, there is no need to program the polynomial syntax: Axes that are programmed with their name and end point only are traversed linearly to their end point. If all axes are programmed in this manner, the control system responds as if G1 were programmed.

Polynomial interpolation is deactivated by another command in the G group (e.g. G0, G1).

**SW 5 and higher**

**Subprogram call POLYPATH:**

With POLYPATH the polynomial interpolation can be specified selectively for the following axis groups:

- **POLYPATH (*AXES*)**
  All path axes and special axes.
- **POLYPATH (*VECT*)** orientation axes (with orientation transformation).

As standard, the programmed polynomials are interpreted as polynomial for both axis groups.

Examples:

**POLYPATH (*VECT*)**

Only the orientation axes are selected for the polynomial interpolation; all other axes are traversed linearly.

**POLPATH ()**

Deactivates the polynomial interpolation for all axes.

**Polynomial coefficient**

The PO value \( [PO(\ldots)] = \ldots = PO(\ldots) \) specifies all polynomial coefficients for an axis. Several values, separated by commas, are specified according to the degree of the polynomial. Different polynomial degrees can be programmed for different axes within one block.
Supplementary conditions

**SW 5 and lower**
- Polynomials for geometry axes/special path axes can only be programmed if either G0/G1 or POLY is active. Therefore, with circular interpolation it is not possible to traverse additional axes via polynomials. As standard, polynomials can only be programmed with PO[... ] if the G code POLY is active.

**SW 5 and higher**
- It is possible to program polynomials without the G code POLY being active. In this case, however, the programmed polynomials are not interpolated; instead the respective programmed endpoint of each axis is approached linearly (G1).
- The polynomial interpolation is then activated by programming POLY.
- Also, if G code POLY is active, with the predefined subprogram POLYPATH (...), you can select which axes are to be interpolated with polynomial.

**SW 6 and higher**
- Coefficients $a_4$ and $a_5$ are only supported by SW 6 and higher.
- New polynomial syntax with PO
  - The syntax used hitherto also remains valid

### Example of applicable polynomial syntax with PO

<table>
<thead>
<tr>
<th>Polynomial syntax used hitherto remains valid</th>
<th>New polynomial syntax (SW 6 and higher)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO[axis identifier]=(.. , ..)</td>
<td>Axis identifier=PO(.. , ..)</td>
</tr>
<tr>
<td>PO[PHI]=(.. , ..)</td>
<td>PHI=PO(.. , ..)</td>
</tr>
<tr>
<td>PO[PSI]=(.. , ..)</td>
<td>PSI=PO(.. , ..)</td>
</tr>
<tr>
<td>PO[THT]=(.. , ..)</td>
<td>THT=PO(.. , ..)</td>
</tr>
<tr>
<td>PO[variable]=IC(.. , ..)</td>
<td>variable=PO IC(.. , ..)</td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved

SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
5.4 Polynomial interpolation – POLY, POLYPATH (SW 5 and higher)

**Programming**

POLY PO[X]=(x_e,a_2,a_3) PO[Y]=(y_e,b_2,b_3) PO[Z]=(z_e,c_2,c_3) PL=n (SW 5 and lower)

POLYPATH ("AXES", "VECT") (SW 5 and higher)

Expansion to polynomials of the 5th degree and new polynomial syntax (SW 6 and higher)

POLY X=PO(x_e,a_2,a_3,a_4,a_5) Y=PO(y_e,b_2,b_3,b_4,b_5) Z=PO(z_e,c_2,c_3,c_4,c_5) PL=n

**Explanation**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLY</td>
<td>Activation of polynomial interpolation with a block containing POLY.</td>
</tr>
<tr>
<td>POLYPATH</td>
<td>Polynomial interpolation can be selected for both the AXIS or VECT axis groups</td>
</tr>
<tr>
<td>PO[axis identifier/variable]=(…,…,…)</td>
<td>End points and polynomial coefficients</td>
</tr>
<tr>
<td>X, Y, Z</td>
<td>Axis name</td>
</tr>
<tr>
<td>x_e, y_e, z_e</td>
<td>Specification of end position for relevant axis; value range as for path dimension</td>
</tr>
<tr>
<td>a_2, a_3, a_4, a_5</td>
<td>Coefficients a_2, a_3, a_4, and a_5 are written with their value; range of values as for path dimension. The last coefficient in each case can be omitted if it equals zero.</td>
</tr>
<tr>
<td>PL</td>
<td>Length of parameter interval over which the polynomials are defined (definition range of function f(p)). The interval always starts at 0. p can be set to values between 0 and PL. Theoretical value range for PL: 0.0001 ... 99999.9999. The PL values applies to the block in which it is programmed. PL=1 is applied if no PL value is programmed.</td>
</tr>
</tbody>
</table>

**Example**

```
N10 G1 X... Y... Z... F600
N11 POLY PO[X]=(1,2.5,0.7) -> Polynomial interpolation ON
  -> PO[Y]=(0.3,1,3.2) PL=1.5
N12 PO[X]=(0,2.5,1.7) PO[Y]=(2.3,1.7) PL=3
... NO PL value programmed; PL=1 applies
N20 M8 H126 ...
N25 X70 PO[Y]=(9.3,1,7.67) PL=5 Mixed settings for axes
N27 PO[X]=(10.2.5) PO[Y]=(2.3) No PL value programmed; PL=1 applies
N30 G1 X... Y... Z. Polynomial interpolation OFF
... 
```
Example of a curve in the X/Y plane

Example:

\[
\begin{align*}
N9 & \ X0 \ Y0 \ G90 \ F100 \\
N10 & \ \text{POLY} \ \text{PO}[Y]=(2) \ \text{PO}[X]=(4,0.25) \ \text{PL}=4
\end{align*}
\]
Special case denominator polynomial

Command \( \text{PO\[\]} = (...) \) can be used to program a common denominator polynomial for the geometry axes (without specification of axes names), i.e. the motion of the geometry axes is then interpolated as the quotient of two polynomials.

With this programming option, it is possible to represent forms such as conics (circle, ellipse, parabola, hyperbola) exactly.

Example

<table>
<thead>
<tr>
<th>POLY G90 X10 Y0 F100</th>
<th>Geometry axes traverse linearly to position X10, Y0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{PO}[X]=(0,–10) ) ( \text{PO}[Y]=(10) ) ( \text{PO}[]=(2,1) )</td>
<td>Geometry axes traverse along quadrant to X0, Y10</td>
</tr>
</tbody>
</table>

The constant coefficient \( (a_0) \) of the denominator polynomial is always assumed to be 1, the specified end point is not dependent on G90/G91.

The result obtained from the above example is as follows:

\[
X(p) = \frac{10(1–p^2)}{1+p^2} \quad \text{and} \quad Y(p) = \frac{20p}{1+p^2}
\]

where \( 0 \leq p \leq 1 \)

As a result of the programmed start points, end points, coefficient \( a_2 \) and \( \text{PL}=1 \), the intermediate values are as follows:

Numerator \( (X)=10+0*p–10p^2 \)
Numerator \( (Y)=0+20*p+0*p^2 \)
Denominator = \( 1+2*p+1*p^2 \)
An alarm is output if a denominator polynomial with zeros is programmed within the interval \([0, PL]\) when polynomial interpolation is active. Denominator polynomials have no effect on the motion of special axes.

**Additional notes**

Tool radius compensation can be activated with G41, G42 in conjunction with polynomial interpolation and can be applied in the same way as in linear or circular interpolation modes.
5.5 Settable path reference, SPATH, UPATH (SW 4.3 and higher)

Programming

SPATH  Path reference for FGROUP axes is length of arc

UPATH  The curve parameter is the path reference for FGROUP axes

Introduction

During polynomial interpolation the user may require two different relationships between the velocity-determining FGROUP axes and the other path axes: The latter are to be controlled

- either synchronized with the path of the FGROUP axes
- or synchronized with the curve parameter.

Previously, only the first motion control variant was implemented; now SW 4.3 and higher offers a G code (SPATH, UPATH) for selecting and programming the desired response.

Function

During polynomial interpolation - and here we are referring to polynomial interpolation in the stricter sense (POLY), all spline interpolation types (ASPLINE, BSPLINE, CSPLINE) and linear interpolation with compressor (COMPON, COMPCURV) - the positions of all path axes \( i \) are preset by means of polynomials \( p_i(U) \). Curve parameter \( U \) moves from 0 to 1 within an NC block, therefore it is standardized.

The axes to which the programmed path feed is to relate can be selected from the path axes by means of language command FGROUP. However, during polynomial interpolation, an interpolation with constant velocity on path \( S \) of these axes usually means a non constant change of curve parameter \( U \).
Therefore, for the axes not contained in FGROUP there are two ways to follow the path:

1. Either they travel synchronized with path S (SPATH)
2. or synchronized with curve parameter U of the FGROUP axes (UPATH).

Both types of path interpolation are used in different applications and can be switched via G codes SPATH and UPATH.

UPATH and SPATH also determine the relationship of the F word polynomial (FPOLY, FCUB, FLIN) with the path movement.

**Example**

The example below shows a square with 20mm side lengths and corners rounded with G643.

The maximum deviations from the exact contour are defined for each axis by the machine data MD 33100: COMPRESS_POS_TOL[...].

```
N10  G1 X... Y... Z... F500
N20  G643 Block-internal corner rounding with G643
N30  X0  Y0
N40  X20 Y0 20mm edge length for axes
N50  X20 Y20
N60  X0  Y20
N70  X0  Y0
N100 M30
```

**Supplementary conditions**

The path reference set is of no importance with

- linear and circular interpolation,
- in thread blocks and
- if all path axes are contained in FGROUP.
5.5 Settable path reference, SPATH, UPATH (SW 4.3 and higher)

Activation

The path reference for the axes that are not contained in FGROUP is set via the two language commands SPATH and UPATH contained in the 45th G code group. The commands are modal. If SPATH is active, the axes are traversed synchronized with the path; if UPATH is active, traversal is synchronized with the curve parameter.

Programming example

The following program example shows the difference between both types of motion control. Both times the default setting FGROUP(X,Y,Z) is active.

```
N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=(10, 10) A10
or
N10 G1 X0 F1000 UPATH
N20 POLY PO[X]=(10, 10) A10
```

In block N20, path S of the FGROUP axes is dependent on the square of curve parameter U. Therefore, different positions arise for synchronized axis A along the path of X, according to whether SPATH or UPATH is active:
Control response at power ON, mode change, Reset, block search, REPOS

After Reset the G code defined via MD 20150: GCODE_RESET_VALUES [44] is active (45th G code group).

The basic setting value for the type of rounding is set in MD 20150: GCODE_RESET_VALUES [9] (10th G code group).

Machine/option data

The G code group value active after Reset is determined via machine data MD 20150: GCODE_RESET_VALUES [44].

In order to maintain compatibility with existing installations, SPATH is set as default value.

The basic setting value for the type of rounding is set in MD 20150: GCODE_RESET_VALUES [9] (10th G code group).

Axial machine data MD 33100: COMPRESS_POS_TOL has been expanded in SW 4.3 and higher: It contains the tolerances for the compressor function and for rounding with G642.
5.6 Measurements with touch trigger probe, MEAS, MEAW

Programming

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEAS=±1</td>
<td>Measurement with probe 1 at measuring input 1</td>
</tr>
<tr>
<td>MEAS=±2</td>
<td>Measurement with probe 2 at measuring input 2</td>
</tr>
<tr>
<td>MEAW=±1</td>
<td>Measurement with probe 1 at measuring input 1</td>
</tr>
<tr>
<td>MEAW=±2</td>
<td>Measurement with probe 2 at measuring input 2</td>
</tr>
</tbody>
</table>

*Max. of two inputs depending on configuration level

Explanation of the commands

- MEAS=±1: G... X... Y... Z... (+1/+2 measurement with deletion of distance-to-go and rising edge)
- MEAS=±2: G... X... Y... Z... (−1/−2 measurement with deletion of distance-to-go and falling edge)
- MEAW=±1: G... X... Y... Z... (+1/+2 measurement without deletion of distance-to-go and rising edge)
- MEAW=±2: G... X... Y... Z... (−1/−2 measurement without deletion of distance-to-go and falling edge)

Sequence

The positions coinciding with the switching edge of the probe are acquired for all axes programmed in the NC block and written for each specific axis to the appropriate memory cell. A maximum of 2 probes can be installed.

Measurement result

The measurement result is available under the following variables for these axes:

- Under $AA_MM[axis]$ in the machine coordinate system
- Under $AA_MW[axis]$ in the workpiece coordinate system

No internal preprocessing stop is generated when these variables are read. A preprocessing stop must be programmed with STOPRE at the appropriate position in the program. The system will otherwise read false values.
5.6 Measurements with touch trigger probe, MEAS, MEAW

**Measuring job status**

Status variable $\text{SAC\_MEA}\_n$ (n= number of probe) can be scanned if the switching state of the touch trigger probe needs to be evaluated in the program:

- 0 Measuring job not performed
- 1 Measuring job successfully completed (probe has switched state)

If the probe is deflected during program execution, this variable is set to 1. At the beginning of a measurement block, the variable is automatically set to correspond to the starting state of the probe.

**Programming measuring blocks, MEAS, MEAW**

When command MEAS is programmed in conjunction with an interpolation mode, actual positions on the workpiece are approached and measured values recorded simultaneously. The distance-to-go between the actual and setpoint positions is deleted.

The MEAW function is employed in the case of special measuring tasks where a programmed position must always be approached.

MEAS and MEAW are programmed in a block with motion commands. The feeds and interpolation types (G0, G1, ...) must be selected to suit the measuring task in hand; this also applies to the number of axes.

**Example:**

N10 MEAS=1 G1 F1000 X100 Y730 Z40

Measurement block with probe at first measuring input and linear interpolation. A preprocessing stop is automatically generated.
5.6 Measurements with touch trigger probe, MEAS, MEAW

Measured value recording
The positions of all path and positioning axes (maximum number of axes depends on control configuration) in the block that have moved are recorded.

In the case of MEAS, the motion is braked in a defined manner after the probe has switched.

Comment
If a GEO axis is programmed in a measurement block, the measured values for all current GEO axes are recorded.
If an axis that participates in a transformation is programmed in a measurement block, the measured values for all axes that participate in this transformation are recorded.

Additional notes
The MEAS and MEAW functions are active non-modally.
5.7 Extended measuring function MEASA, MEAWA, MEAC
(SW 4 and higher, option)

Programming

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEASA[axis]=(mode, TE1,…, TE 4)</td>
<td>Measurement with delete distance-to-go</td>
</tr>
<tr>
<td>MEAWA[Achse]=(Modus, TE 1,…, TE 4)</td>
<td>Measurement without delete distance-to-go</td>
</tr>
<tr>
<td>MEAC[axis]=(mode, measurement memory, TE 1,…TE 4)</td>
<td>Continuous measurement without deletion of distance-to-go</td>
</tr>
</tbody>
</table>

Explanation

**Axis**
Name of channel axis used for measurement

**Mode**
Two-digit setting for operating mode consisting of Measuring mode (ones decade) and

0  | Cancel measurement task
1  | Mode 1: Up to 4 trigger events that can be activated simultaneously
2  | Mode 2: Up to 4 trigger events that can be activated sequentially
3  | Mode 3: Up to 4 trigger events that can be activated sequentially

Note: Mode 3 not possible with MEAC

**Measuring system** (tens’ decade)

0  | or no setting: Active measuring system
1  | Measuring system 1
2  | Measuring system 2
3  | Both measuring systems

**TE 1…4**
Trigger event

1  | Rising edge, probe 1
-1 | Falling edge, probe 1
2  | Rising edge, probe 2
-2 | Falling edge, probe 2

**Measurement memory**
Number of FIFO (circulating storage)
Function

Axial measurement is available from SW 4. With this system, measurements can be taken axially with several probes and several measuring systems.

When MEASA, MEAWA is programmed, up to four measured values are acquired for the programmed axis in each measuring run and stored in system variables in accordance with the trigger event. MEASA and MEAWA are non-modal commands.

Continuous measuring operations can be executed with MEAC. In this case, the measurement results are stored in FIFO variables. The maximum number of measured values per measuring run is also 4 with MEAC.

Sequence

The measurements can be programmed in the parts program or from a synchronized action (Chapter 10). Please note that only one measuring job can be active at any given time for each axis.

Additional notes

- The feed must be adjusted to suit the measuring task in hand.
- In the case of MEASA and MEAWA, the correctness of results can be guaranteed only at feedrates with which no more than one trigger event of the same type and no more than 4 trigger events occur in each position controller cycle.
- In the case of continuous measurement with MEAC, the ratio between the interpolation cycle and position control cycle must not exceed 8:1.
Trigger events
A trigger event comprises the number of the probe and the trigger criterion (rising or falling edge) of the measuring signal.

Up to 4 trigger events of the addressed probe can be processed for each measurement, i.e. up to two probes with two measuring signal edges each. The processing sequence and the maximum number of trigger events depends on the selected mode.

The same trigger event is only permitted to be programmed once in a measuring job (only applies to mode 1)!

Operating mode
The first digit in the mode setting selects the desired measuring system. If only one measuring system is installed, but a second programmed, the installed system is automatically selected.

With the second digit, i.e. the measurement mode, measuring process is adapted to the capabilities of the connected control system:
- **Mode 1**: Trigger events are evaluated in the chronological sequence in which they occur. When this mode is selected, only one trigger event can be programmed for six-axis modules. If more than one trigger event is specified, the mode selection is switched automatically to mode 2 (without message).
- **Mode 2**: Trigger events are evaluated in the programmed sequence.
- **Mode 3**: Trigger events are evaluated in the programmed sequence, however no monitoring of trigger event 1 at START.

Additional notes
No more than 2 trigger events can be programmed if 2 measuring systems are in use.
Measurement with and without delete distance-to-go

When command MEASA is programmed, the distance-to-go is not deleted until all required measured values have been recorded.

The MEAWA function is employed in the case of special measuring tasks where a programmed position must always be approached.

MEASA and MEAWA can be programmed in the same block.

If MEASA/MEAWA is programmed with MEAS/MEAW in the same block, an error message is output.

- MEASA cannot be programmed in synchronized actions.
  - As an alternative, MEAWA plus the deletion of distance-to-go can be programmed as a synchronized action.
- If the measuring job with MEAWA is started from the synchronized actions, the measured values will only be available in machine coordinates.

Measurement results for MEASA, MEAWA

The results of measurements are available under the following system variables:

- In machine coordinate system:
  - $AA_{_MM}1[axis]$ Measured value of programmed measuring system on trigger event 1
  - $AA_{_MM}4[axis]$ Measured value of programmed measuring system on trigger event 4
- In workpiece coordinate system:
  - $AA_{_WM}1[axis]$ Measured value of programmed measuring system on trigger event 1
  - $AA_{_WM}4[axis]$ Measured value of programmed measuring system on trigger event 4
Additional notes

No internal preprocessing stop is generated when these variables are read.
A preprocessing stop must be programmed with STOPRE (Section 15.1) at the appropriate position. False values will otherwise be read in.

If axial measurement is to be started for a geometry axis, the same measuring job must be programmed explicitly for all remaining geometry axes.
The same applies to axes involved in a transformation.
Example:
N10 MEASA[Z]=(1,1) MEASA[Y]=(1,1) MEASA[X]=(1,1) G0 Z100;
or
N10 MEASA[Z]=(1,1) POS[Z]=100

Measuring job with two measuring systems

If a measuring job is executed by two measuring systems, each of the two possible trigger events of both measuring systems of the relevant axis is acquired. The assignment of the reserved variables is therefore preset:

$AA_MM1[axis]$ or $AA_MW1[axis]$ Measured value of measuring system 1 on trigger event 1
$AA_MM2[axis]$ or $AA_MW2[axis]$ Measured value of measuring system 2 on trigger event 1
$AA_MM3[axis]$ or $AA_MW3[axis]$ Measured value of measuring system 2 on trigger event 1
$AA_MM4[axis]$ or $AA_MW4[axis]$ Measured value of measuring system 2 on trigger event

Measuring probe status can be read via
$A_PROBE[n]$
n=Probe
1==Probe deflected
0==Probe not deflected
Measuring job status for MEASA, MEAWA
If the probe switching state needs to be evaluated in the program, then the measuring job status can be interrogated via $AC_MEA[n]$, with $n =$ number of probe.
Once all the trigger events of probe "n" that are programmed in a block have occurred, this variable switches to the “1” stage. Its value is otherwise 0.

If measuring is started from synchronized actions, $AC_MEA$ is not updated. In this case, new PLC status signals DB(31-48) DDB62 bit 3 or the equivalent variable $AA_MEAACT["Axis"]$ must be interrogated.
Meaning: $AA_MEAACT == 1$: Measuring active
$AA_MEAACT == 0$: Measuring not active
References: /FB/ M5, Measurement

Continuous measurement MEAC
The measured values for MEAC are available in the machine coordinate system and stored in the programmed FIFO[n] memory (circulating memory). If two probes are configured for the measurement, the measured values of the second probe are stored separately in the FIFO[n+1] memory configured especially for this purpose (defined in machine data). The FIFO memory is a circulating memory in which measured values are written to $AC_FIFO$ variables according to the circulation principle.
References: /PGA/ Chapter 10, Synchronized Actions

Additional notes
- FIFO contents can be read only once from the circulating storage. If these measured data are to be used multiply, they must be buffered in user data.
- If the number of measured values for the FIFO memory exceeds the maximum value defined in machine data, the measurement is automatically terminated.
- An endless measuring process can be implemented by reading out measured values cyclically. In this case, data must be read out at the same frequency as new measured values are read in.
### Programming example

**Measurement with delete distance-to-go in mode 1**

*(evaluation in chronological sequence)*

#### a) with one measuring system

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100</td>
<td>MEASA[X] = (1,1,-1) G01 X100 F100</td>
<td>Measurement in mode 1 with active measuring system. Wait for measuring signal with rising/falling edge from probe 1 on travel path to X = 100.</td>
</tr>
<tr>
<td>N110</td>
<td>STOPRE</td>
<td>Preprocessing stop</td>
</tr>
<tr>
<td>N120</td>
<td>IF $AC_MEA[1] == FALSE gotof END</td>
<td>Check success of measurement.</td>
</tr>
<tr>
<td>N130</td>
<td>R10 = $AA_MM1[X]</td>
<td>Store measured value acquired on first programmed trigger event (rising edge)</td>
</tr>
<tr>
<td>N140</td>
<td>R11 = $AA_MM2[X]</td>
<td>Store measured value acquired on second programmed trigger event (falling edge)</td>
</tr>
</tbody>
</table>

#### Programming example

**b) with two measuring systems**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N200</td>
<td>MEASA[X] = (31,1,-1) G01 X100 F100</td>
<td>Measurement in mode 1 with both measuring systems. Wait for measuring signal with rising/falling edge from probe 1 on travel path to X = 100.</td>
</tr>
<tr>
<td>N210</td>
<td>STOPRE</td>
<td>Preprocessing stop</td>
</tr>
<tr>
<td>N230</td>
<td>R10 = $AA_MM1[X]</td>
<td>Store measured value of measuring system 1 on rising edge</td>
</tr>
<tr>
<td>N240</td>
<td>R11 = $AA_MM2[X]</td>
<td>Store measured value of measuring system 2 on rising edge</td>
</tr>
<tr>
<td>N250</td>
<td>R12 = $AA_MM3[X]</td>
<td>Store measured value of measuring system 1 on falling edge</td>
</tr>
</tbody>
</table>
5.7 Extended measuring function MEASA, MEAWA, MEAC

N260 R13 = $AA_MM4[X]
Store measured value of measuring system 2 on falling edge

N270 END:

Measurement with delete distance-to-go in mode 2
(evaluation in programmed sequence)

... N100 MEASA[X] = (2,1,-1,2,-2) G01 X100 F100
Measurement in mode 2 with active measuring system. Wait for measuring signal in the following order: Rising edge of probe 1, falling edge of probe 1, rising edge of probe 2, falling edge of probe 2, on travel path to X = 100.

N110 STOPRE
Preprocessing stop

N120 IF $AC_MEA[1] == FALSE goto PROBE2
Check success of measurement with probe 1

PROBE2

N130 R10 = $AA_MM1[X]
Store measured value acquired on first programmed trigger event (rising edge probe 1)

N140 R11 = $AA_MM2[X]
Store measured value acquired on second programmed trigger event (rising edge probe 1)

N150 PROBE2:

Check success of measurement with probe 2

N170 R12 = $AA_MM3[X]
Store measured value acquired on third programmed trigger event (rising edge probe 2)

N180 R13 = $AA_MM4[X]
Store measured value acquired on fourth programmed trigger event (rising edge probe 2)

N190 END:
Programming example

Continuous measurement in mode 1
(evaluation in chronological sequence)

Measurement of up to 100 measured values

\[
\begin{align*}
N110 & \text{DEF REAL MEASVALUE[100]} \\
N120 & \text{DEF INT loop = 0} \\
N130 & \text{MEAC[X] = (1,1,-1) G01 X1000 F100} \\
\end{align*}
\]

Measure in mode 1 with active measuring system, store measured values under $\text{AC_FIFO1}$, wait for measuring signal with falling edge from probe 1 on travel path to $X = 1000$.

\[
\begin{align*}
N135 & \text{STOPRE} \\
N140 & \text{MEAC[X] = (0)} \\
\end{align*}
\]

Terminate measurement when axis position is reached.

\[
\begin{align*}
N150 & \text{R1 = $\text{AC_FIFO1}[4]$} \\
N160 & \text{FOR loop = 0 TO R1-1} \\
N170 & \text{MEASVALUE[loop] = $\text{AC_FIFO1}[0]$} \\
\end{align*}
\]

Read measured values from $\text{AC_FIFO1}$ and store.

\[
\begin{align*}
N180 & \text{ENDFOR} \\
\end{align*}
\]

Measurement with delete distance-to-go after ten measured values

\[
\begin{align*}
(\times) & \text{Delete distance-to-go} \\
N20 & \text{MEAC[x]=(1,1,1,-1) G01 X1000 F500} \\
N30 & \text{MEAC[X]=(0)} \\
N40 & \text{R1 = $\text{AC_FIFO1}[4]$} \\
\end{align*}
\]

Number of measured values
5.7 Extended measuring function MEASA, MEAWA, MEAC

The following programming errors are detected and indicated appropriately:

- **MEASA/MEAWA is programmed with MEAS/MEAW in the same block**
  
  Example:
  
  `N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100`

- **MEASA/MEAWA with number of parameters <2 or >5**
  
  Example:
  
  `N01 MEAWA[X]=(1) G01 F100 POS[X]=100`

- **MEASA/MEAWA with trigger event not equal to 1/ -1/ 2/ -2**
  
  Example:
  
  `N01 MEASA[B]=(1,1,3) B100`

- **MEASA/MEAWA with invalid mode**
  
  Example:
  
  `N01 MEAWA[B]=(4,1) B100`

- **MEASA/MEAWA with trigger event programmed twice**
  
  Example:
  
  `N01 MEASA[B]=(1,1,-1,2,-1) B100`

- **MEASA/MEAWA and missing GEO axis**
  
  Example:
  
  `N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50 Y50 Z50 F100 GEO axis X/Y/Z`

- **Inconsistent measuring job with GEO axes**
  
  Example:
  
  `N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2) G01 X50 Y50 Z50 F100`
5.8 Special functions for OEM users

**OEM addresses**
The meaning of OEM addresses is determined by the OEM user.
Their functionality is incorporated by means of compile cycles. Five OEM addresses are reserved.
The address identifiers are settable.
OEM addresses can be programmed in any block.

**OEM interpol**
The OEM user can define two additional interpolations. Their functionality is incorporated by means of compile cycles.
The names of G functions (OEMIPO1, OEMIPO2) are set by the OEM user.
**OEM addresses (see above) can be used specifically for OEM interpolations.**

**Reserved G groups G800–819**
Two G groups with ten OEM G functions each are reserved for OEM users.
These allow the functions incorporated by an OEM user to be accessed for external applications.

**Functions and subprograms**
OEM users can also set up predefined functions and subprograms with parameter transfer.
5.9 Programmable motion end criterion (SW 5.1 and higher)

Programming

- FINEA[<axis>]
- COARSEA[<axis>]
- IPOENDA[<axis>]
- IPOBRKA(<axis>[, [value as percentage]])
- ADISPOSA(<axis>, [<int>],[<real>])

Multiple specifications are possible

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINEA</td>
<td>Motion end when &quot;Exact stop FINE&quot; reached</td>
</tr>
<tr>
<td>COARSEA</td>
<td>Motion end when &quot;Exact stop COARSE&quot; reached</td>
</tr>
<tr>
<td>IPOENDA</td>
<td>Motion end when &quot;Interpolator-Stop&quot; reached</td>
</tr>
<tr>
<td>IPOBRKA</td>
<td>Block change in braking ramp possible (SW 6.2 and higher)</td>
</tr>
<tr>
<td>ADISPOSA</td>
<td>Size of tolerance window for end of motion criterion (SW 6.4 and higher)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Axis</th>
<th>Channel axis name (X, Y, ...)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value as percentage</td>
<td>When relative to the braking ramp of the block change should be as %</td>
</tr>
<tr>
<td>Int</td>
<td>Mode 0: tolerance window not active</td>
</tr>
<tr>
<td></td>
<td>1: tolerance window relative to setpoint position</td>
</tr>
<tr>
<td></td>
<td>2: tolerance window relative to actual position</td>
</tr>
<tr>
<td>Real</td>
<td>Size of tolerance window. This value is entered in setting data 43610: ADISPOSA_VALUE synchronized with the main run</td>
</tr>
</tbody>
</table>

Function

Similar to the block change criterion for continuous-path interpolation (G601, G602 and G603), the end of motion criterion can be programmed in a parts program for single axis interpolation or in synchronized action for the command/PLC axes. Depending on the end of motion criterion set, parts program blocks or technology cycle blocks with single axis motion take different times to complete. The same applies for PLC positioning statements via FC15/16/18.
System variable $AA_MOTEND

The default motion end characteristic can be requested via system variable $AA_MOTEND[<axis>].

- $AA_MOTEND[<axis>] = 1
  Motion end with "Exact stop fine"
- $AA_MOTEND[<axis>] = 2
  Motion end with "Exact stop coarse"
- $AA_MOTEND[<axis>] = 3
  End of motion with "IPO-Stop".
- $AA_MOTEND[<axis>] = 4 (SW 6.2 and higher)
  Block change criterion braking ramp of axis motion
- $AA_MOTEND[<axis>] = 5 (SW 6.4 and higher)
  Block change in braking ramp with tolerance window relative to "setpoint position".
- $AA_MOTEND[<axis>] = 6 (SW 6.4 and higher)
  Block change in braking ramp with tolerance window relative to "actual position".

Additional notes

The last programmed value is retained after RESET.

References:
/FB1/, V1 Feedrates

SW 6.2 and higher

Block change criterion in braking ramp

The percentage value is entered in SD 43600: IPOBRAKE_BLOCK_EXCHANGE synchronized with the main run. If no value is specified, the current value of this setting data is effective.
The range is adjustable from 0% to 100%.

Additional tolerance window for IPOBRKA

SW 6.4 and higher, an additional block change criterion tolerance window can be selected as well as the existing block change criterion in the braking ramp. Release only occurs when the axis
- as before, reaches the preset %-value of its braking ramp and
- SW 6.4 and higher, its current actual or setpoint position is no further than a tolerance from the end of the axis in the block.

For more information on the block change criterion of the positioning axes, please refer to:

References:
/FB2/, P2 positioning axes
/PG/, Feed rate control and spindle motion
5.9 Programmable motion end criterion (SW 5.1 and higher)

Programming examples

...  
N110 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]
Traversing to position X100 with a path velocity of 1000rpm, an acceleration value of 90% and end of motion on reaching the interpolator stop

...  
Traversing to position X50 when input 1 is active, with a path velocity of 2000rpm, an acceleration value of 140% and end of motion on reaching the interpolator stop

Block change criterion braking ramp in the parts program

; default setting active
N40 POS[X]=100
; block change occurs when X-axis reaches position 100 and fine exact stop
N20 IPOBRKA(X,100) ; activate block change criterion braking ramp
N30 POS[X]=200 ; block change occurs as soon as X-axis starts to brake
N40 POS[X]=250
; the x-axis does not brake at position 200 but continues to position 250,
; as soon as the X-axis starts to brake, the block change occurs
N50 POS[X]=0 ; the X-axis brakes and moves back to position 0
; the block change occurs at position 0 and fine exact stop
N60 X10F100
N70 M30
...

Block change criterion braking ramp within synchronized actions

Within the technology cycle:

FINEA ; end of motion criterion fine exact stop
POS[X]=100 ; technology cycle block change occurs when X-axis has reached position 100 and fine exact stop
IPOBRKA(X,100) ; activate block change criterion braking ramp
POS[X]=100 ; POS[X]=100; technology cycle block change occurs,
; as soon as the X-axis starts to brake
POS[X]=250 ; the X-axis does not brake at position 200 but continues
; to position 250, as soon as the X-axis starts to brake
; the block change in the technology cycle occurs
POS[X]=250 ; the X-axis brakes and moves back to position 0
; the block change occurs at position 0 and fine exact stop
M17
5.10 Programmable servo parameter block (SW 5.1 and higher)

Programming

SCPARA[<Axis>]= <Value>

Explanation of the commands

<table>
<thead>
<tr>
<th>SCPARA</th>
<th>Define parameter block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>Channel axis name (X, Y, ...)</td>
</tr>
<tr>
<td>Value</td>
<td>Desired parameter block (1&lt;= value &lt;=6)</td>
</tr>
</tbody>
</table>

Function

Using SCPARA, it is possible to program the parameter block (consisting of MDs) in the parts program and in synchronized actions (previously only via PLC).

DB3n DBB9 Bit3
To prevent conflicts between the PLC–user request and NC–user request, a further bit is defined on the PLC–>NCK interface:

DB3n DBB9 Bit3 "Parameter block definition locked through SCPARA".

In the case of a locked parameter block for SCPARA, an error message is produced if programmed.

The current parameter block can be polled using the system variables $AA_SCPAR[<Axis>].
Additional notes

- Up to SW 5.1, the servo-parameter block can be specified only by the PLC (DB3n DBB9 Bit0–2). For G33, G331 and G332, the most suitable parameter block is selected by the control.
- If the servo parameter block is to be changed both in a parts program and in a synchronized action and the PLC, the PLC application program must be extended.
- References: /FB1/V1 Feedrates

Programming example

```
N110 SCPARA[X]= 3  The 3rd parameter block is selected for axis X
```

...
5.10 Programmable servo parameter block (SW 5.1 and higher)
Frames

6.1 Coordinate transformation via frame variables ............................................... 6-236
6.2 Frame variables/assigning values to frames ......................................................... 6-241
6.3 Coarse/fine offset .................................................................................................. 6-248
6.4 DRF offset ............................................................................................................ 6-249
6.5 External zero offset .............................................................................................. 6-250
6.6 Programming PRESET offset, PRESETON .......................................................... 6-251
6.7 Deactivating frames ............................................................................................. 6-252
6.8 Frame calculation from 3 measuring points in the area: MEAFRAME .............. 6-253
6.9 NCU-global frames (SW 5 and higher) ................................................................. 6-256
   6.9.1 Channel-specific frames .................................................................................. 6-257
   6.9.2 Frames active in the channel ......................................................................... 6-259
6.1 Coordinate transformation via frame variables

Definition of coordinate transformation with frame variables

In addition to the programming options already described in the Programming Guide "Fundamentals", you can also define coordinate systems with predefined frame variables.

Coordinate systems
The following coordinate systems are defined:

- **MCS**: Machine coordinate system
- **BCS**: Basic coordinate system
- **BOS**: Basic origin system
- **SZS**: Settable zero system
- **WCS**: Workpiece coordinate system

What is a predefined frame variable?
Predefined frame variables are vocabulary words whose use and effect are already defined in the control language and which can be processed in the NC program.
Possible frame variable:
- Base frame (basic offset)
- Settable frames
- Programmable frame
Frame variable/frame relationship
A coordinate transformation can be activated by assigning the value of a frame to a frame variable.

Example: $P_PFRAME=CTRANS(X,10)$

Frame variable:
$P_PFRAME$ means: current programmable frame.

Frame:
$CTRANS(X,10)$ means: programmable zero offset of X axis by 10 mm.

Reading out actual values
The current actual values of the coordinate system can be read out via predefined variables in the parts program:

- $AA_IM[axis]$ Read actual value in MCS
- $AA_IB[axis]$ Read actual value in BCS
- $AA_IBN[axis]$ Read actual value in BOS
- $AA_IEN[axis]$ Read actual value in SZS
- $AA_IW[axis]$ Read actual value in WCS

Overview of predefined variables

$P_BFRAME$
Current base frame variable that establishes the reference between the basic coordinate system (BCS) and the basic origin system (BOS).

For the base frame described via $P_UBFR$ to be immediately active in the program, either
- you have to program a G500, G54...G599, or
- you have to describe $P_BFRAME$ with $P_UBFR$. 
### 6.1 Coordinate transformation via frame variables

**$P_IFRAME**

Current, settable frame variable that establishes the reference between the basic origin system (BOS) and the settable zero system (SZS).

- **$P_IFRAME** corresponds to $P_UIFR[$P_IFRNUM]

After G54 is programmed, for example, $P_IFRAME contains the translation, rotation, scaling and mirroring defined by G54.

**$P_PFRAME**

Current, programmable frame variable that establishes the reference between the settable zero system (SZS) and the workpiece coordinate system (WCS).

- $P_PFRAME contains the frame resulting from the programming of TRANS/ATRANS, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR or the assignment of CTRANS, CROT, CMIRROR, CSCALE to the programmable FRAME.

**$P_ACTFRAME**

Current total frame resulting from chaining of the current base frame variable $P_BFRAME, the current settable frame variable $P_IFRAME and the current programmable frame variable $P_PFRAME.

- **$P_ACTFRAME** describes the currently valid workpiece zero.

- If $P_IFRAME, $P_BFRAME or $P_PFRAME are changed, $P_ACTFRAME is recalculated.

- **$P_ACTFRAME** corresponds to $P_BFRAME;$P_IFRAME;$P_PFRAME
6.1 Coordinate transformation via frame variables

Base frame and settable frame are effective after Reset if MD 20110
RESET_MODE_MASK is set as follows:
Bit0=1, bit14=1 --> $P_UBFR$ (base frame) effective
Bit0=1, bit5=1 --> $P_UIFR$ [$P_UIFRNUM$] (settable frame) effective

Predefined settable frames $P_UBFR$
The base frame is programmed with $P_UBFR$, but it is not simultaneously active in the parts program. The base frame programmed with $P_UBFR$ is included in the calculation if
- Reset was activated and bits 0 and 14 are set in MD RESET_MODE_MASK and
- instructions G500, G54...G599 were executed.

Predefined settable frames $P_UIFR[n]$
The predefined frame variable $P_UIFR[n]$ can be used to read or write the settable zero offsets G54 to G599 from the parts program.

These variables produce a one-dimensional array of type FRAME called $P_UIFR[n]$.

Assignment to G commands
Five predefined settable frames are set as standard $P_UIFR[0]$...$P_UIFR[4]$ or 5 G commands with the same meaning – G500 and G54 to G57 – at whose addresses values can be stored.
Frames

6.1 Coordinate transformation via frame variables

$P_IFRAME$ = $P_UIFR[0]$ corresponds to G500
$P_IFRAME$ = $P_UIFR[1]$ corresponds to G54
$P_IFRAME$ = $P_UIFR[2]$ corresponds to G55
$P_IFRAME$ = $P_UIFR[3]$ corresponds to G56
$P_IFRAME$ = $P_UIFR[4]$ corresponds to G57

You can change the number of frames with machine data:
$P_IFRAME$ = $P_UIFR[5]$ corresponds to G505
... ... ...
$P_IFRAME$ = $P_UIFR[99]$ corresponds to G599

This allows you to generate up to 100 coordinate systems which can be called up globally in different programs, for example, as zero point for various fixtures.

Frame variables must be programmed in a separate NC block in the NC program.
Exception: programming of a settable frame with G54, G55, ...

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
6.2 Frame variables/assigning values to frames

Values can be assigned directly, frames can be chained or frames can be assigned to other frames in the NC program.

Direct value assignment

Programming

$P_PFRAME=CTRANS (X, axis value, Y, axis value, Z, axis value, ...)
$P_PFRAME=CROT (X, angle, Y, angle, Z, angle, ...)
$P_PFRAME=CSCALE (X, scale, Y, scale, Z, scale, ...)
$P_PFRAME=CMIRROR (X, Y, Z)

Programming $P_BFRAME is carried out analog to $P_PFRAME.

Explaination of the commands

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRANS</td>
<td>Translation of specified axes</td>
</tr>
<tr>
<td>CROT</td>
<td>Rotation around specified axes</td>
</tr>
<tr>
<td>CSCALE</td>
<td>Scale change on specified axes</td>
</tr>
<tr>
<td>CMIRROR</td>
<td>Direction reversal on specified axis</td>
</tr>
</tbody>
</table>

Function

You can use these functions to assign frames or frame variables directly in the NC program.

Sequence

You can program several arithmetic rules in succession.

Example:

$P_PFRAME=CTRANS (...) :CROT (...) :CSCALE...

Please note that the commands must be connected by the colon chain operator: (...)(...).
This causes the commands firstly to be linked and secondly to be executed additively in the programmed sequence.
Additional notes

The values programmed with the above commands are assigned to the frames and stored.

The values are not activated until they are assigned to the frame of an active frame variable $P_BFRAME or $P_PFRAME.

Programming example

Translation, rotation and mirroring are activated by value assignment to the current programmable frame.

N10 $P_PFRAME=CTRANS(X,10,Y,20,Z,5):CROT(Z,45):CMIRROR(Y)
6.2 Frame variables/assigning values to frames

Reading and changing frame components

Programming (examples)

R10=$P_UIFR[$P_UIFRNUM, X, RT]
Assign the angle of rotation RT around the X axis from currently valid settable zero offset $P_UIFRNUM to the variable R10.

R12=$P_UIFR[25, Z, TR]
Assign the offset value TR in Z from the data record of set frame no. 25 to the variable R12.

R15=$P_PFRAME[Y, TR]
Assign the offset value TR in Y of the current programmable frame to the variable R15.

$P_PFRAME[X, TR]=25
Modify the offset value TR in X of the current programmable frame. X25 applies immediately.

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_UIFRNUM</td>
<td>This command automatically establishes the reference to the currently valid settable zero offset.</td>
</tr>
<tr>
<td>$P_UIFR[n, ...]</td>
<td>Specify the frame number n to access the settable frame no. n.</td>
</tr>
<tr>
<td>TR</td>
<td>Specify the component to be read or modified: TR translation, FI translation fine, RT rotation, SC scale change, MI mirroring.</td>
</tr>
<tr>
<td>MI</td>
<td>The corresponding axis is also specified (see examples).</td>
</tr>
</tbody>
</table>
Function

This feature allows you to access individual data of a frame, e.g. a specific offset value or angle of rotation.

You can modify these values or assign them to another variable.

Sequence

Calling frame

By specifying the system variable $P_UIFRNUM you can access the current zero offset set with $P_UIFR or G54, G55, ... ($P_UIFRNUM contains the number of the currently set frame).

All other stored settable $P_UIFR frames are called up by specifying the appropriate number $P_UIFR[n].

For predefined frame variables and user-defined frames, specify the name, e.g. $P_IFRAME.

Calling data

The axis name and the frame component of the value you want to access or modify are written in square brackets, e.g. [X, RT] or [Z, MI].
6.2 Frame variables/assigning values to frames

Linking complete frames
A complete frame can be assigned to another frame.

Programming (examples)
DEF FRAME SETTING1
SETTING1=CTRANS(X,10)
$P_PFRA ME=SETTING1

DEF FRAME SETTING4
SETTING4=$P_PFRA ME
$P_PFRA ME=SETTING4

Assign the values of the user frame
SETTING1 to the current programmable
frame.

The current programmable frame is
stored temporarily and can be recalled.

Additional notes
Value range for RT rotation
Rotation around 1st geometry axis: –180° to +180°
Rotation around 2nd geometry axis: –89.999° to +90°
Rotation around 3rd geometry axis: –180° to +180°

Frame chaining
Programming (examples)

$P_IFRAME=$P_UIFR[15]:$P_UIFR[16]


$P_UIFR[15] contains, for example,
data for zero offsets. The data of
$P_UIFR[16], e.g. data for rotations,
are subsequently processed additively.

The settable frame 3 is created by
chaining the settable frames 4 and 5.
6.2 Frame variables/assigning values to frames

Function

Frame chaining is suitable for the description of several workpieces, arranged on a pallet, which are to be machined in the same process.

Sequence

The frames are chained in the programmed sequence. The frame components (translations, rotations, etc.) are executed additively in succession.

The frame components can only contain intermediate values for the description of pallet tasks. These are chained to generate various workpiece zeroes.

Please note that the frames must be linked to one another by the colon chain operator :.
6.2 Frame variables/assigning values to frames

Definition of new frames

Programming

```
DEF FRAME PALLET1

PALLET1=CTRANS(...) : CROT(...) ...
```

Function

In addition to the predefined settable frames described above, you also have the option of creating new frames. This is achieved by creating variables of type FRAME to which you can assign a name of your choice.

Sequence

You can use the functions CTRANS, CROT, CSSCALE and CMIRROR to assign values to your frames in the NC program. You will find more information on this subject on the previous pages.

Frame rotation definition

Function

Frame rotations can be used to define application-specific orientations in the area:

- **ROT**: Individual rotations for all geometry axes
- **ROTS, AROTS, CROTS**: Rotation by specifying a solid angle (max. 2); see description in /FB1/ K2: coordinate systems.
- **TOFRAME**: Rotation by frame "TOFRAME", with Z axis pointing in the tool direction.
- **TOROT**: Rotation by frame "TOROT", which only overwrites the rotation component of frames that have already been programmed.
6.3 Coarse/fine offset

**Function**

**Fine offset**
A fine offset of the base frames and of all other settable frames can be programmed with command `CFINE(X, .., Y, ..)`.

**Coarse offset**
The coarse offset is defined with `CTRANS(...)`.

Coarse and fine offset add up to the total offset.

**Programming**

```plaintext
$P_{UBFR}=CTRANS(x, 10) : CFINE(x, 0.1) : CROT(x, 45) ; chaining offset, fine offset and rotation

$P_{UIFR}[1]=CFINE(x, 0.5, y, 1.0, z, 0.1) ; the total frame is overwritten with CFINE, incl. coarse offset.
```

Access to the individual components of the fine offset is achieved through component specification Fi.

**Programming**

```plaintext
DEF REAL FINEX ; Definition of variable FINEX
FINEX=$P_{UIFR}[F_{UIFRNUM}, x, Fi] ; Readout the fine offset via variable FINEX
FINEX=$P_{UIFR}[3, X, Fi] ; Readout the fine offset of X axis in the 3rd frame via variable FINEX
```

Fine offset can only take place if MD 18600:
`MM_FRAME_FINE_TRANS=1`

A fine offset changed via operator input is only active after the corresponding frame is activated, i.e. activation is conducted via G500, G54...G599. An activated fine offset of a frame is active for as long as the frame is active.
The programmable frame has no fine offset. If the programmable frame is assigned a frame with fine offset, then the total offset is established by adding the coarse and the fine offset. When reading the programmable frame the fine offset is always zero.

Machine manufacturer

SW 5 and higher
The fine offset can be configured by means of MD 18600 MM_FRAME_FINE_TRANS in the following variants:
0: Fine offset cannot be entered or programmed. G58 and G59 are not possible.
1: Fine offset for settable frames, base frames, programmable frames, G58 and G59 can be entered/programmed

6.4 DRF offset

Offset using handwheel, DRF
In addition to all the translations described in this section, you can also define zero offsets with the handwheel (DRF offset).

The DRF offset acts on the basic coordinate system. See the diagram for the relationships.

You will find more information in the Operator’s Guide.

Clear DRF offset, DRFOF
DRFOF clears the handwheel offset for all axes assigned to the channel. DRFOF is programmed in a separate NC block.
6.5 External zero offset

**External zero offset**

This is another way of moving the zero point between the basic and workpiece coordinate system.

Only linear translations can be programmed with the external zero offset.

**Programming offset values, $AA\_ETRANS**

The offset values are programmed by assigning the axis-specific system variables.

**Assigning offset value**

$AA\_ETRANS[axis] = R_i$

$R_i$ is the arithmetic variable of type REAL which contains the new value.

The external offset is generally set by the PLC and not specified in the parts program.

The value entered in the parts program only becomes active when the corresponding signal is enabled at the VDI interface (NCU-PLC interface).
6.6 Programming PRESET offset, PRESETON

Programming

PRESETON(AXIS, VALUE, ...)

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESETON</td>
<td>Set actual value</td>
</tr>
<tr>
<td>Axis</td>
<td>Machine axis parameter</td>
</tr>
<tr>
<td>Value</td>
<td>New actual value to apply to the specified axis</td>
</tr>
</tbody>
</table>

Function

In special applications, it can be necessary to assign a new programmed actual value to one or more axes at the current position (stationary).

Note: Preset mode with synchronized actions should only be implemented the vocabulary word "WHEN" or "EVERY".

Sequence

The actual values are assigned to the machine coordinate system – the values refer to the machine axes.

Example:

N10 G0 A760
N20 PRESETON(A1, 60)

Axis A travels to position 760. At position 760, machine axis A1 is assigned the new actual value 60.
From this point, positioning is performed in the new actual value system.

The reference point becomes invalid with the function PRESETON. You should therefore only use this function for axes which do not require a reference point. If the original system is to be restored, the reference point must be approached with G74 – see Section 3.1.
6.7 Deactivating frames

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRFOF</td>
<td>Deactivate (clear) the handwheel offsets (DRF)</td>
</tr>
<tr>
<td>G53</td>
<td>Non-modal deactivation of programmable and all settable frames</td>
</tr>
<tr>
<td>G153</td>
<td>Non-modal deactivation of programmable frames, base frames and all settable frames</td>
</tr>
<tr>
<td>SUPA</td>
<td>Non-modal deactivation of all programmable frames, base frames, all settable frames and handwheel offsets (DRF)</td>
</tr>
</tbody>
</table>

Additional notes

The programmable frames are cleared by assigning a "zero frame" (without axis specification) to the programmable frame.

Example:

$P_PFRAME=TRANS( )
$P_PFRAME=ROT( )
$P_PFRAME=SCALE( )
$P_PFRAME=MIRROR( )
6.8 Frame calculation from three measuring points in the area: MEAFRAME

MEAFRAME is an extension of the 840D language used for supporting measuring cycles. This function is valid in SW 4.3 and higher.

Function

When a workpiece is positioned for machining, its position relative to the Cartesian machine coordinate system is generally both shifted and rotated referring to its ideal position. For exact machining or measuring either a costly physical adjustment of the part is required or the motions defined in the parts program must be changed.

A frame can be determined by probing three points in the area for which the ideal positions are known. Probing is performed with a tactile or optical sensor touching special holes or spheres that are precisely fixed to the backing plate.

The function MEAFRAME calculates the frame from three ideal and the corresponding measured points. In order to map the measured coordinates onto the ideal coordinates using a rotation and a translation, the triangle formed by the measured points must be congruent to the ideal triangle. This is achieved by means of a compensation algorithm that minimizes the sum of squared deviations needed to reshape the measured triangle into the ideal triangle. Since the effective distortion can be used to judge the quality of the measurement, MEAFRAME returns it as an additional variable.
### Programming

MEAFRAME(IDEAL_POINT, MEAS_POINT, FIT_QUALITY)

### Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEAFRAME</td>
<td>Frame calculation of three measured points in space</td>
</tr>
<tr>
<td>IDEAL_POINT</td>
<td>2-dim. array of real data containing the three coordinates of the ideal points</td>
</tr>
<tr>
<td>MEAS_POINT</td>
<td>2-dim. array of real data containing the three coordinates of the measured points</td>
</tr>
<tr>
<td>FIT_QUALITY</td>
<td>Variable of type real returning the following information:</td>
</tr>
<tr>
<td></td>
<td>-1: The ideal points are located approximately on a straight line: The frame could not be calculated. The frame variable returned contains a neutral frame.</td>
</tr>
<tr>
<td></td>
<td>-2: The measured points are located approximately on a straight line: The frame could not be calculated. The frame variable returned contains a neutral frame.</td>
</tr>
<tr>
<td></td>
<td>-4: The calculation of the rotation matrix failed for a different reason</td>
</tr>
</tbody>
</table>

**Positive value:**

Sum of the distortions (distances between the points) needed to reshape the measured triangle into one that is congruent to the ideal triangle.

### Application example

```plaintext
; Parts program 1
;
DEF FRAME CORR_FRAME
;
; Setting measured points
DEF REAL IDEAL_POINT[3,3] = SET(10.0,0.0,0.0, 0.0,10.0,0.0, 0.0,0.0,10.0)
DEF REAL MEAS_POINT[3,3] = SET(10.1,0.2,-0.2, -0.2,10.2,0.1, -0.2,0.2, 9.8); for test
DEF REAL FIT_QUALITY = 0
 ;
DEF REAL ROT_FRAME_LIMIT = 5; allows max. 5° rotation of the part position
DEF REAL FIT_QUALITY_LIMIT = 3; allows max. 3 mm distortion between the ideal; and the measured triangle
DEF REAL SHOW_MCS_POS1[3]
DEF REAL SHOW_MCS_POS2[3]
DEF REAL SHOW_MCS_POS3[3]
;
```
N100 G01 G90 F5000
N110 X0 Y0 Z0
;
N200 CORR_FRAME=MEAFRAME(IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
;
N230 IF FIT_QUALITY < 0
SETAL(65000)
GOTOF NO_FRAME
ENDIF
;
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT
SETAL(65010)
GOTOF NO_FRAME
ENDIF
;
N250 IF CORR_FRAME[X,RT] > ROT_FRAME_LIMIT; limiting the 1st RPY angle
SETAL(65020)
GOTOF NO_FRAME
ENDIF
;
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT; limiting the 2nd RPY angle
SETAL(65021)
GOTOF NO_FRAME
ENDIF
;
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT; limiting the 3rd RPY angle
SETAL(65022)
GOTOF NO_FRAME
ENDIF
;
N300 $P_IFRAME=CORR_FRAME; activate the probe frame via a settable frame
;
; check the frame by positioning the geometry axes at the ideal points
;
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1] Z=IDEAL_POINT[0,2]
N410 SHOW_MCS_POS1[0]=$AA_IM[X]
N420 SHOW_MCS_POS1[1]=$AA_IM[Y]
;
N500 X=IDEAL_POINT[1,0] Y=IDEAL_POINT[1,1] Z=IDEAL_POINT[1,2]
N510 SHOW_MCS_POS2[0]=$AA_IM[X]
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]
;
N600 X=IDEAL_POINT[2,0] Y=IDEAL_POINT[2,1] Z=IDEAL_POINT[2,2]
N610 SHOW_MCS_POS3[0]=$AA_IM[X]
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]
;
Deactivate settable frame, as preset with zero frame (no value set)

\[ \text{NO\_FRAME:} \]
\[ \text{M0} \]
\[ \text{M30} \]

### 6.9 NCU-global frames (SW 5 and higher)

#### Function

NCU-global frames are only available once for all channels of each NCU. NCU-global frames can be written and read from all channels. The NCU-global frames are activated in the respective channel. Channel axes and machine axes with offsets can be scaled and mirrored by means of global frames. With global frames there is no geometrical relationship between the axes. Therefore, it is not possible to perform rotations or program geometry axis identifiers.

- It is not possible to use global frames for rotations. Programming a rotation is refused and alarm: "18310 channel %1 block %2 frame: rotation not allowed" is displayed.
- Chaining of global frames and channel-specific frames is possible. The resulting frame contains all frame elements including rotations for all axes. If a frame with rotation elements is assigned to a global frame, it is rejected and alarm "Frame: rotation not allowed" is displayed.

**NCU-global base frames: $P\_NCBFR[n]**

You can configure up to 8 NCU-global basic frames.
Machine manufacturer

The number of global base frames is configured via machine data. (See /FB/ K2, Axes, Coordinate Systems, Frames)
Channel-specific base frames can be present at the same time.

Global frames can be written and read from all channels of an NCU. When writing global frames, the user must pay attention to channel coordination, for example, by using Wait marks (WAITMC).

NCU-global settable frames: $P_{UIFR}[n]
All settable frames G500, G54...G599 can be configured either NCU-global or channel-specific.

Machine manufacturer

All settable frames can be reconfigured as global frames via MD 18601 MM_NUM_GLOBAL_USER_FRAMES.
See /FB/ K2, Axes, Coordinate Systems, Frames.
Channel axis identifiers and machine axis identifiers can be used as axis identifiers for the frame program commands. Programming of geometry identifiers is rejected with an alarm.

6.9.1 Channel-specific frames

Function

The number of base frames can be configured in the channel via MD 28081 MM_NUM_BASE_FRAMES.
The standard configuration provides at least one base frame per channel. A maximum of eight base frames are supported per channel. In addition to the eight base frames, there can also be eight NCU-global base frames in the channel.
Settable frames/base frames can be written and read from the control and the PLC

- via the parts program and
- via the OPI.

Fine offset is also possible for global frames.

Suppression of global frames also takes place, as is the case with channel-specific frames, via G53, G153, SUPA and G500.

$\text{SP\_CHBFR}[n]$

The base frames can be read and written via system variable $\text{SP\_CHBFR}[n]$. When writing a base frame, the chained total base frame is not activated; it is only activated when the G500, G54..G599 instruction is executed. The variable mainly serves as memory for writing processes to the MMC and PLC base frame. These frame variables are saved by data backup.

**First basic frame in the channel**

Writing to a predefined variable $\text{SP\_UBFR}$ will not activate the basic frame with array index 0 simultaneously, but it will be activated only after a G500, G54..G599 command is executed. The variable can also be written and read in the program.

$\text{SP\_UBFR}$

$\text{SP\_UBFR}$ is identical to $\text{SP\_CHBFR}[0]$. As standard, there is always a base frame in the channel making the system variable compatible with older versions. If there is no channel-specific base frame, an alarm is issued at read/write: "Frame: instruction not permissible".
6.9.2 Frames active in the channel

Function

SW 6.1 and higher
Current system frames for
$P\_\text{PARTFRAME}$ TCARR and PAROT
$P\_\text{SETFRAME}$ preset actual value memory and scratching,
$P\_\text{EXTFRAME}$ zero offset external,
You can read and write the current system frame in the parts program via these system variables.

$P\_\text{NCBFRAME}[n]$
Current NCU-global basic frames
You can read and write the current global basic frame field elements via system variable $P\_\text{NCBFRAME}[n]$. The resulting total base frame is calculated by means of the write process in the channel.

The modified frame is only active in the channel in which the frame was programmed. If the frame is to be changed for all channels of an NCU, both $[n]$ and $P\_\text{NCBFRAME}[n]$ have to be programmed. The other channels must then still activate the frame with, for example, G54. When writing a base frame, the total base frame is calculated again.

$P\_\text{CHBFRAME}[n]$
Current channel basic frames
You can read and write the current channel basic frame field elements via system variable $P\_\text{CHBFRAME}[n]$. The resulting total base frame is calculated by means of the write process in the channel. When writing a base frame, the total base frame is calculated again.
$P\_BFRAME

Current first basic frame in the channel

You can read and write the current basic frame with array index 0, which applies for the channel, in the parts program via the predefined frame variable $P\_BFRAME. The written basic frame is immediately included in the calculation.

$P\_BFRAME$ is identical to $P\_CHIFRAME[0]$. The default is for the system variable to always have a valid value. If there is no channel-specific base frame, an alarm is issued at read/write: "Frame: instruction not permissible".

$P\_ACTBFRAME

Total basic frame

Variable $P\_ACTBFRAME$ determines the chained total basic frame. The variable can only be read.

$P\_ACTBFRAME$ corresponds to $P\_NCIFRAME[0] : ... : P\_NCIFRAME[n] : P\_CHIFRAME[0] : ... : P\_CHIFRAME[n].$
You can specify with $P_{CHBFRMASK}$ which channel-specific base frames, and with $P_{NCBFRMASK}$ which NCU-global base frames, are to be included in the calculation.

By programming the variables the total base frame and the total frame are calculated again. After a Reset is performed, the basic setting value is

$P_{CHBFRMASK} = MC_{CHBFRAME\_RESET\_MASK}$ and
$P_{NCBFRMASK} = MN_{NCBFRAME\_RESET\_MASK}$.

E.g.

$P_{NCBFRMASK} = 'H81' ; P_{NCBFRAME}[0] : P_{NCBFRAME}[7]$
$P_{CHBFRMASK} = 'H11' ; P_{CHBFRAME}[0] : P_{CHBFRAME}[4]$

$P_{IFRAME}$

Current settable frame

You can read and write the current settable frame, which applies in the channel, in the parts program via the predefined frame variable $P_{IFRAME}$. The written settable frame is immediately included in the calculation.

With NCU-global settable frames, the modified frame is only active in the channel in which the frame was programmed. If the frame is to be changed for all channels of an NCU, both $P_{UIFR}[n]$ and $P_{IFRAME}$ have to be programmed. The other channels must then still activate the respective frame with, for example, G54.

SW 6.1 and higher

Current system frames for

$P\_TOOLFRAME$ TOROT and TOFRAME

SW 6.3 and higher

$P\_WPFRAME$ Workpiece reference points

You can read and write the current system frame in the parts program via these system variables.
$P\_PFRAME

**Current programmable frame**

$P\_PFRAME$ is the programmable frame which results from programming TRANS/ATRANS, G58/G59, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR or from assigning CTRANS, CROT, CMIRROR, CScale to the programmable frame.

Current, programmable frame variable that establishes the reference between the settable zero system (SZS) and the workpiece coordinate system (WCS).

**SW 6.3 and higher**

**Current system frame for $P\_CYCFRAME$ Cycles**

You can read and write the current system frame in the parts program via this system variable.

$P\_ACTFRAME

**Current total frame**

The current resulting total frame $P\_ACTFRAME$ now results from chaining all basic frames, the current settable frame and the programmable frame.

The current frame is always updated if a frame element is modified.

**SW 6.3 and higher, $P\_ACTFRAME$ corresponds to**

$P\_SETFRAME : P\_EXTFRAME : P\_PARTFRAME : P\_ACTBFRAME :

$P\_IFRAME : P\_TOOLFRAME : P\_WPFRAME : P\_PFRAME : P\_CYCFRAME

**SW 6.4 and higher, $P\_ACTFRAME$ corresponds to**

$P\_PARTFRAME : P\_SETFRAME : P\_EXTFRAME : P\_ACTBFRAME :

$P\_IFRAME : P\_TOOLFRAME : P\_WPFRAME : P\_PFRAME : P\_CYCFRAME
6.9 NCU-global frames (SW 5 and higher)

Frame chaining

The current frame consists of the total basic frame, the settable frame, the system frame and the programmable frame according to the current total frame mentioned above.
6.9 NCU-global frames (SW 5 and higher)

840D 840D 810D 840Di
NCU 571 NCU 572 NCU 573
Transformations

7.1 Three, four and five axis transformation: TRAORI ................................................................. 7-266
  7.1.1 Programming tool orientation .......................................................................................... 7-269
  7.1.2 Orientation axes reference – ORIWCS, ORIMCS ......................................................... 7-274
  7.1.3 Singular positions and how to handle them ................................................................. 7-275
  7.1.4 Orientation axes (SW 5.2 and higher) ....................................................................... 7-276
  7.1.5 Cartesian PTP travel (from SW 5.2) ......................................................................... 7-279
  7.1.6 Online tool length compensation (SW 6.4 and higher) ........................................ 7-284

7.2 Milling turned parts: TRANSMIT ....................................................................................... 7-287

7.3 Cylinder surface transformation: TRACYL ....................................................................... 7-290

7.4 Inclined axis: TRAANG ..................................................................................................... 7-296
  7.4.1 Inclined axis programming: G05, G07 (SW 5.3 and higher) ................................... 7-300

7.5 Constraints when selecting a transformation ................................................................. 7-302

7.6 Deselect transformation: TRAFOOF ............................................................................... 7-304

7.7 Chained transformations .................................................................................................. 7-305

7.8 Switchable geometry axes, GEOAX .................................................................................. 7-308
7.1 Three, four and five axis transformation: TRAORI

To obtain optimum cutting conditions when machining surfaces with a three-dimensional curve, it must be possible to vary the setting angle of the tool.

The machine design to achieve this is stored in the axis data.

Cardanic tool head

Three linear axes (X, Y, Z) and two orientation axes define the setting angle and the operating point of the tool here. One of the two orientation axes is created as an inclined axis, in our example A’ - in many cases, placed at 45°.

The axis sequence of the orientation axes and the orientation direction of the tool are set up via the machine data subject to the machine kinematics. In the examples shown here, you can see the arrangements in the CA machine kinematics example!
There are the following possible relationships:
A' is below angle $\varphi$ to the X axis
B' is below angle $\varphi$ to the Y axis
C' is below angle $\varphi$ to the Z axis

Angle $\varphi$ can be configured in the range 0° to +89° via machine data.

Depending on the orientation direction selected for the tool, the active working plane (G17, G18, G19) must be set in the NC program in such a way that tool length compensation works in the direction of tool orientation.

**Transformation with a swiveling linear axis**
This is an arrangement with a moving workpiece and a moving tool.
The kinematics consists of three linear axes (X, Y, Z) and two orthogonally arranged rotary axes.
The first rotary axis is moved, for example, over a compound slide of two linear axes, the tool standing parallel to the third linear axis.
The second rotary axis turns the workpiece.
The third linear axis (swivel axis) lies in the compound slide plane.

The axis sequence of the rotary axes and the orientation direction of the tool are set up via the machine data subject to the machine kinematics.

There are the following possible relationships:

**Axes:**
- 1st rotary axis
- 2nd rotary axis
- Swiveled linear axis

**Axis sequences:**
- A A B B C C
- B C A C A B
- Z Y Z X Y X
3-axis and 4-axis transformations
3-axis and 4-axis transformations are special forms of 5-axis transformations. The user can configure two or three translatory axes and one rotary axis. The transformations assume that the rotary axis is orthogonal on the orientation plane. Tool orientation is only possible in the plane that is perpendicular to the rotary axis. Transformation supports machine types with a mobile tool and a mobile workpiece.

Configuration and programming for 3-axis and 4-axis transformations are the same as for 5-axis transformations.

Programming
TRAORI(n)
TRAFOOF

Explanation of the commands

- **TRAORI** Activates the first specified orientation transformation
- **TRAORI(n)** Activates the orientation transformation specified by n
- **n** The number of the transformation (n = 1 or 2), TRAORI(1) corresponds to TRAORI
- **TRAFOOF** Disable transformation

Additional notes
When the transformation is enabled, the positional data (X, Y, Z) always relates to the tip of the tool.

Changing the position of the rotary axes involved in the transformation causes so many compensating movements of the remaining machine axes that the position of the tool tip is unchanged.
7.1 Three, four and five axis transformation: TRAORI

7.1.1 Programming tool orientation

5-axis programs are usually generated by CAD/CAM systems and not entered at the control. So the following explanations are directed mainly at the programmers of postprocessors.

There are three options available when programming tool orientation:

1. Programming the motion of the rotary axes. The change of orientation always occurs in the basic or machine coordinate system. The orientation axes are traversed as synchronized axes.

2. Programming in Euler angles or RPY angles via A2, B2, C2 or Programming the direction vector via A3, B3, C3. The direction vector points from the tool tip towards the toolholder.

3. Programming via the lead angle LEAD and the tilt angle TILT (face milling).

In all cases, orientation programming is only permissible if an orientation transformation is active.

Advantage: These programs can be transferred to any machine kinematics.
7.1 Three, four and five axis transformation: TRAORI

Programming

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1 X Y Z A B C</td>
<td>Programming the motion of the rotary axes.</td>
</tr>
<tr>
<td>G1 X Y Z A2= B2= C2=</td>
<td>Programming in Euler angles</td>
</tr>
<tr>
<td>G1 X Y Z A3= B3= C3=</td>
<td>Programming the direction vector</td>
</tr>
<tr>
<td>G1 X Y Z A4= B4= C4=</td>
<td>Programming the surface normal vector at block start</td>
</tr>
<tr>
<td>G1 X Y Z A5= B5= C5=</td>
<td>Programming the surface normal vector at end of block</td>
</tr>
<tr>
<td>LEAD</td>
<td>Lead angle for programming tool orientation</td>
</tr>
<tr>
<td>TILT</td>
<td>Tilt angle for programming tool orientation</td>
</tr>
</tbody>
</table>

Machine data can be used to switch between Euler and RPY angles.

Programming in Euler angles

The values programmed during orientation programming with A2, B2, C2 are interpreted as Euler angles (in degrees).

The orientation vector results from turning a vector in the Z direction firstly with A2 around the Z axis, then with B2 around the new X axis and lastly with C2 around the new Z axis.

In this case the value of C2 (rotation around the new Z axis) is meaningless and does not have to be programmed.
Programming in RPY angles

The values programmed during orientation programming with A2, B2, C2 are interpreted as RPY angles (in degrees).

The orientation vector results from turning a vector in the Z direction firstly with C2 around the Z axis, then with B2 around the new Y axis and lastly with A2 around the new X axis.

In contrast to Euler angle programming, all three values here have an effect on the orientation vector.

Programming the direction vector

The components of the direction vector are programmed with A3, B3, C3. The vector points towards the toolholder; the length of the vector is meaningless.

Vector components that have not been programmed are set equal to zero.
Face milling

Face milling is used to machine curved surfaces of any kind.

For this type of 3D milling, you require line-by-line definition of 3D paths on the workpiece surface. The tool shape and dimensions are taken into account in the calculations that are normally performed in CAM. The fully calculated NC blocks are then read into the control via postprocessors.

Surface description

The path curvature is described by surface normal vectors with the following components:
- A4, B4, C4 start vector at block start
- A5, B5, C5 end vector at block end

If a block only contains the start vector, the surface normal vector will remain constant throughout the block.

If a block only contains the end vector, interpolation will run from the end value of the previous block via large circle interpolation to the programmed end value.

If both start and end vectors are programmed, interpolation runs between the two directions, also via large circle interpolation. This allows continuously smooth paths to be created.

In the initial setting, surface normal vectors – whatever the active G17 to G19 level – point in the Z direction.

The length of a vector is meaningless.

Vector components that have not been programmed are set to zero.
With active ORIWCS (see following pages), the surface normal vectors relate to the active frame and also turn when the frame is turned.

The surface normal vector must be perpendicular to the path tangent, within a limit value set via machine data, otherwise an alarm will be output.

**Programming the tool orientation with LEAD and TILT**

The resultant tool orientation is determined from:
- the path tangent,
- the surface normal vector
- the lead angle LEAD
- the tilt angle TILT at end of block.

**Explanation of the commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAD</td>
<td>Angle relative to the surface normal vector in the plane put up by the path tangent and the surface normal vector</td>
</tr>
<tr>
<td>TILT</td>
<td>Angle in the plane, perpendicular to the path tangent relative to the surface normal vector</td>
</tr>
</tbody>
</table>

**Behavior at inside corners (for 3D-tool compensation)**

If the block at an inside corner is shortened, the resultant tool orientation is also achieved at end of block.
7.1.2 Orientation axes reference – ORIWCS, ORIMCS

Programming

N. ORIMCS
or
N. ORIWCS

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIMCS</td>
<td>Rotation in the machine coordinate system</td>
</tr>
<tr>
<td>ORIWCS</td>
<td>Rotation in the workpiece coordinate system</td>
</tr>
</tbody>
</table>

Function

With orientation programming in the workpiece coordinate system via Euler or RPY angles or the orientation vector, ORIMCS/ORIWCS can be used to adjust the course of the rotary motion.

Sequence

With ORIMCS, the movement executed by the tool is dependent on the machine kinematics. With an orientation change with a fixed tool tip, interpolation between the rotary axis positions is linear.

With ORIWCS, the tool movement is not dependent on the machine kinematics. With an orientation change with a fixed tool tip, the tool moves in the plane set up by the start and end vectors.
Additional notes

ORIWCS is the basic setting. If it is not immediately obvious with a 5-axis program which machine it should run on, always choose ORIWCS. Which movements the machine actually executes depend on the machine kinematics.

With ORIMCS, you can program actual machine movements, for example, to avoid collisions with devices, etc.

Machine data $MC_ORI_IPO_WITH_G_CODE specifies the active interpolation mode: ORIMCS/ORIWCS or ORIMACHAX/ORIVIRTAX (see Subsection 7.1.4).

7.1.3 Singular positions and how to handle them

Notes on ORIWCS:

Orientation movements in the singular setting area of the 5-axis machine require vast movements of the machine axes. (For example, with a rotary swivel head with C as the rotary axis and A as the swivel axis, all positions with A = 0 are singular.)

To avoid overloading the machine axes, the velocity control vastly reduces the tool path velocity near the singular positions.

With machine data
$MC_TRAFO5_NON_POLE_LIMIT
$MC_TRAFO5_POLE_LIMIT
the transformation can be parameterized in such a way that orientation movements close to the pole are put through the pole and rapid machining is possible.

Note on SW 5.2:

As from SW5.2, singular positions will only be handled by MD $MC_TRAFO5_POLE_LIMIT (see Description of Functions Part 3, Subsection 2.8.4).
7.1.4 Orientation axes (SW 5.2 and higher)

Programming

N.. ORIEULER or ORIRPY
or
N.. ORIVIRT1 or ORIVIRT2
N.. G1 X Y Z A2= B2= C2=

Explanation of the commands

ORIEULER Orientation programming using Euler angles
ORIRPY Orientation programming using RPY angles
ORIVIRT1 Orientation programming using virtual orientation axes
   (definition 1), definition according to MD $MC_ORIAX_TURN_TAB_1
ORIVIRT2 Orientation programming using virtual orientation axes
   (definition 2), definition according to MD $MC_ORIAX_TURN_TAB_2
G1 X Y Z A2= B2= C2= Angle programming of virtual axes

Programming

N.. ORIAXES or ORIVECT
N.. G1 X Y Z A B C

Explanation of the commands

ORIAXES Linear interpolation of orientation axes
ORIVECT Large circle interpolation
ORIMCS Rotation in the machine coordinate system
   For description, see Subsection 7.1.2
ORIWCS Rotation in the workpiece coordinate system
   For description, see Subsection 7.1.2
G1 X Y Z A B C Programming the machine axis position

Function

The orientation axis function describes the orientation of the tool in space. This introduces an additional third degree of freedom that describes the rotation around itself. This is necessary for 6-axis transformations.
MD $MC\_ORI\_DEF\_WITH\_G\_CODE$ specifies how the programmed angles A2, B2, C2 are defined:
The definition is made according to MD $MC\_ORIENTATION\_IS\_EULER$ (default) or
the definition is made according to G_group 50 (ORIEULER, ORIRPY, ORIVIRT1, ORIVIRT2).

MD $MC\_ORI\_IPO\_WITH\_G\_CODE$ specifies which interpolation mode is active:
ORIWCS/ORIMCS or ORIAXES/ORIVECT.

**JOG mode**
Interpolation for orientation angles in this mode of operation is always linear. During continuous and incremental traversal via the traversing keys, only one orientation axis can be traversed. Orientation axes can be traversed simultaneously using the handwheels.

For manual travel of the orientation axes, the channel-specific feed override switch or the rapid traverse override switch work at rapid traverse override.

A separate velocity setting is possible with the following machine data:
$MC\_JOG\_VELO\_RAPID\_GEO$
$MC\_JOG\_VELO\_GEO$
$MC\_JOG\_VELO\_RAPID\_ORI$
$MC\_JOG\_VELO\_ORI$

**SW 6.3 and higher**
In JOG mode, the cartesian manual travel function can, for SINUMERIK 840D with the "Handling transformation package" and for Sinumerik 810D powerline from SW 6.1 set up separately the translation of the geometry axes in the reference systems MCS, WCS and TCS.

**Reference notes:**
SINUMERIK 840D/FM-NC Description of Functions (Part 3), "Handling transformation package".
/FB/ F2, 3-axis to 5-axis transformations
Feed programming

<table>
<thead>
<tr>
<th>FORI1</th>
<th>Feed for swiveling the orientation vector on the large circle</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORI2</td>
<td>Feed for the overlaid rotation around the swiveled orientation vector</td>
</tr>
</tbody>
</table>

With orientation movements, the programmable feed corresponds to an angular velocity [degrees/min].

**Effectiveness of the feed via G code:**

When programming ORIAXES, the feed for an orientation axis can be limited via the FL instruction (feed limit).

When programming ORIVECT, the feed must be programmed with FORI1 or FORI2. FORI1 and FORI2 must only be programmed once in the NC block. Traversal always takes the shortest path during this programming.

The smallest feed always operates for the overlaid motion of turning and swiveling. With orientation movements, the feed corresponds to an angular velocity [degrees/min].

If geometry axes and orientation axes traverse a common path, the traversing movement is determined from the smallest feed.
7.1.5 Cartesian PTP travel (from SW 5.2)

Programming

N.. TRAORI
N.. STAT=`B10` TU=`B100` PTP
N.. CP

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PTP</strong></td>
<td>Point to Point (point to point movement)</td>
</tr>
<tr>
<td></td>
<td>The movement is executed as a synchronized axis movement; the slowest axis involved in the movement is the dominating axis for the velocity.</td>
</tr>
<tr>
<td><strong>CP</strong></td>
<td>Continuous path (path motion)</td>
</tr>
<tr>
<td></td>
<td>The movement is executed as cartesian path motion</td>
</tr>
<tr>
<td><strong>STAT</strong></td>
<td>Position of the articulated joints; this value is dependent on the transformation.</td>
</tr>
<tr>
<td><strong>TU</strong></td>
<td>TURN information</td>
</tr>
<tr>
<td></td>
<td>This makes it possible to clearly approach axis angles between -360 degrees and +360 degrees.</td>
</tr>
</tbody>
</table>

Function

This function can be used to program a position in a cartesian coordinate system, however, the movement of the machine occurs in the machine coordinates.

The function can be used, for example, when changing the position of the articulated joint, if the movement runs through a singularity.

Note:

The function is only useful in conjunction with an active transformation. Furthermore, "PTP travel" is only permissible in conjunction with G0 and G1.

Sequence

The commands PTP and CP effect the changeover between cartesian traversal and traversing the machine axes. These are modal. CP is the default setting.
7 Transformations

7.1 Three, four and five axis transformation: TRAORI

Programming the position (STAT=)
A machine position is not uniquely determined just by positional data with cartesian coordinates and the orientation of the tool. Depending on the kinematics involved, there can by as many as eight different and crucial articulated joint positions. These are specific to the transformation. To be able to uniquely convert a cartesian position into the axis angle, the position of the articulated joints must be specified with the command STAT=. The "STAT" command contains a bit for each of the possible positions as a binary value.

Reference notes:
The various transformations are included in the document:
SINUMERIK 840D/FM-NC Description of Functions (Part 3), "Handling transformation package".

The positional bits to be programmed for "STAT" are included in the document:
SINUMERIK 840D/FM-NC Description of Functions (Part 3), "3-axis to 5-axis transformation".

Programming the axis angle (TU=)
To be able to clearly approach by axis angles < ±360 degrees, this information must be programmed using the command "TU=".
The command is non-modal.

The axes traverse by the shortest path:
• when no TU is programmed for a position
• with axes that have a traversing range > ±360 degrees
Example:
The target position shown in the diagram can be approached in the negative or positive direction. The direction is programmed under the address A1.  
A1=225°, TU=bit 0, → positive direction  
A1=−135°, TU=bit 1, → negative direction

Smoothing between CP and PTP motion

A programmable transition rounding between the blocks is possible with G641.  
The size of the rounding area is the path in mm or inch, from which or to which the block transition is to be rounded. The size must be specified as follows:
• for G0 blocks with ADISPOS
• for all the other motion commands with ADIS.

The path calculation corresponds to considering of the F addresses for non-G0 blocks. The feed is kept to the axes specified in FGROUP(...).

Feed calculation:
For CP blocks, the cartesian axes of the basic coordinate system are used for the calculation.  
For PTP blocks, the corresponding axes of the machine coordinate system are used for the calculation.
Additional notes

Mode change
The "Cartesian PTP travel" function is only useful in the AUTO and MDA modes of operation. When changing the mode to JOG, the current setting is retained.
When the G code PTP is set, the axes will traverse in MCS. When the G code CP is set, the axes will traverse in WCS.

Power On / Reset
After a power ON or after a Reset, the setting is dependent on the machine data $MC_GCODE_RESET_VALUES[48]$. The default traversal mode setting is "CP".

Repositioning
If the function "Cartesian PTP travel" was set during the interruption block, PTP can also be used for repositioning.

Overlaid movements
DRF offset or external zero offset are only possible to a limited extent in cartesian PTP travel. When changing from PTP to CP motion, there must be no overrides in the BCS.
### Programming example

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>G0 X0 Y-30 Z60 A-30 F10000</td>
<td>Starting position → Elbow up</td>
</tr>
<tr>
<td>N20</td>
<td>TRAORI(1)</td>
<td>Transformation ON</td>
</tr>
<tr>
<td>N30</td>
<td>X1000 Y0 Z400 A0</td>
<td></td>
</tr>
<tr>
<td>N40</td>
<td>X1000 Z500 A0 STAT=<code>B10</code> TU=<code>B100</code> PTP</td>
<td>Reorientation without transformation → Elbow down</td>
</tr>
<tr>
<td>N50</td>
<td>X1200 Z400 CP</td>
<td>Transformation active again</td>
</tr>
<tr>
<td>N60</td>
<td>X1000 Z500 A20</td>
<td></td>
</tr>
<tr>
<td>N70</td>
<td>M30</td>
<td></td>
</tr>
</tbody>
</table>
7.1.6 Online tool length compensation (SW 6.4 and higher)

Programming

N.. TRAORI
N.. TOFFON(X,25)
N.. WHEN TRUE DO $AA_TOFF[X]

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOFFON</td>
<td>Tool Offset ON (activate online tool length compensation)</td>
</tr>
<tr>
<td></td>
<td>When activating, an offset value can be specified for the relevant direction of compensation and this is immediately recovered.</td>
</tr>
<tr>
<td>TOFFOF</td>
<td>Tool Offset OF (reset online tool length compensation)</td>
</tr>
<tr>
<td></td>
<td>The relevant compensation values are reset and a preprocessing stop is initiated.</td>
</tr>
</tbody>
</table>

Function

Use the system variable $AA_TOFF[ ] to overlay the effective tool lengths in accordance with the three tool directions three-dimensionally in real time. The three geometry axis identifiers are used as the index. This defines the number of active directions of compensation by the geometry axes active at the same time. All the overrides can be active simultaneously.

Application

The online tool length compensation function can be used for:

- orientation transformation TRAORI
- orientable toolholder TCARR

Note:

Online tool length compensation is an option, that first has to be enabled. This function is only useful in conjunction with an active orientation transformation or an active orientable toolholder.
Additional notes

Block preparation
During block preparation in preprocessing, the current tool length offset active in the main run is also taken into consideration. To allow extensive use to be made of the maximum permissible axis velocity, it is necessary to stop block preparation with a STOPRE preprocessing stop while a tool offset is set up. The tool offset is then always known at the time of preprocessing if tool length compensations can no longer be changed after program startup, or if, following a change to the tool length compensations, more blocks were executed than the IPO buffer between preprocessing and main run can accept.

Variable $AA_TOFF_PREP_DIFF
The size of the difference between the current compensation active in the interpolator and the compensation active at the time the block was prepared, can be queried in the $AA_TOFF_PREP_DIFF[ ] variable.

Adjusting machine data and setting data
The following machine data is available for online tool length compensation:

- MD 20610: ADD_MOVE_ACCEL_RESERVE Reserve for velocity planning
- MD 21190: TOFF_MODE The content of the system variable $AA_TOFF[ ] is recovered or integrated as an absolute value.
- MD 21194: TOFF_VELO Velocity of the online tool length compensation
- MD 21196: TOFF_ACCEL Acceleration of the online tool length compensation

Setting data for presetting limit values
- SD 42970: TOFF_LIMIT Upper limit of the tool length compensation value
Programming example

Tool length compensation selection

<table>
<thead>
<tr>
<th>MD 21190: TOFF_MODE = 1</th>
<th>; Absolute values are approached</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD 21194: TOFF_VELO[0] = 1000</td>
<td></td>
</tr>
<tr>
<td>MD 21196: TOFF_VELO[1] = 1000</td>
<td></td>
</tr>
<tr>
<td>MD 21194: TOFF_VELO[2] = 1000</td>
<td></td>
</tr>
<tr>
<td>MD 21196: TOFF_ACCEL[0] = 1</td>
<td></td>
</tr>
<tr>
<td>MD 21196: TOFF_ACCEL[1] = 1</td>
<td></td>
</tr>
<tr>
<td>MD 21196: TOFF_ACCEL[2] = 1</td>
<td></td>
</tr>
</tbody>
</table>

N5       DEF REAL XOFFSET
N10      TRAORI(1) ; Transformation ON
N20      TOFFON(Z) ; Activation of online tool length offset for the Z tool direction
N30      WHEN TRUE DO $AA_TOFF[Z] = 10 G4 F5 ; For the Z tool direction, a tool length offset of 10 is interpolated
... 
N40      TOFFON(X) ; Activation of online tool length offset for the X tool direction
N50      ID=1 DO $AA_TOFF[X] = $AA_IW[X2] G4 F5 ; For the X tool direction, an offset is executed subject to the position of axis X2
... 
N100     XOFFSET = $AA_TOFF_VAL[X] ; Assign current offset in X direction
N120     TOFFON(X, -XOFFSET) G4 F5 ; For the X tool direction, the tool length offset will be returned to 0 again

Tool length compensation deselection

N10      TRAORI(1) ; Transformation ON
N20      TOFFON(X) ; Activating the Z tool direction
N30      WHEN TRUE DO $AA_TOFF[X] = 10 G4 F5 ; For the X tool direction, a tool length offset of 10 is interpolated
... 
N80      TOFFOF(X) ; Positional offset of the X tool direction is deleted: ...$AA_TOFF[X] = 0
; No axis is traversed
; To the current position in WCS, the positional offset is added in accordance with the current orientation

References

/FB/ F2, 3-axis to 5-axis transformations
7.2 Milling turned parts: TRANSMIT

Programming

TRANSMIT or TRANSMIT(n)
TRAFOOF

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSMIT</td>
<td>Activates the first specified Transmit function</td>
</tr>
<tr>
<td>TRANSMIT(n)</td>
<td>Activates the n-th specified Transmit function; the maximum for n is 2</td>
</tr>
<tr>
<td></td>
<td>(TRANSMIT(1) corresponds to TRANSMIT).</td>
</tr>
<tr>
<td>TRAFOOF</td>
<td>Disables an active transformation</td>
</tr>
</tbody>
</table>

An active TRANSMIT transformation is also disabled if one of the remaining transformations is activated in the particular channel (e.g. TRACYL, TRAANG, TRAORI).

The TRANSMIT function facilitates the following performance:

- Machining the end face of turned parts clamped for turning (holes, contours).
- A cartesian coordinate system can be used to program this machining.
- The control transforms the programmed traversing movements of the cartesian coordinate system to the traversing movements of the real machine axes (default situation):
  - Rotary axis
  - Infeed axis perpendicular to the rotary axis
  - Longitudinal axis parallel to the rotary axis
  - The linear axes are positioned perpendicular to one another.
- Tool center offset relative to the turning center is permissible.
- The velocity control considers the limitations defined for rotary motion.
Rotary axis
The rotary axis cannot be programmed, as it is assigned by a geometry axis and is thus not directly programmable as a channel axis.

Pole
SW 3.x and lower
Traversing through the pole (the origin of the cartesian coordinate system) is prevented. A movement that runs through the pole stops at the pole and an alarm is output. With milling center offset, movement correspondingly stays at the edge of the area not to be approached.

SW 4 and higher
There are two options for traversing through the pole:
1. Traverse only the linear axis
2. Traverse to the pole, rotate the rotary axis at the pole and traveling away from the pole
Make the selection using MD 24911 and 24951.

References
/FB/ M1 Kinematic transformations
Programming example

N10 T1 D1 G54 G17 G90 F5000 G94  
Tool selection
N20 G0 X20 Z10 SPOS=45  
Approach initial position
N30 TRANSMIT  
Activate the Transmit function
N40 ROT RPL=-45
N50 ATRANS X-2 Y10  
Adjust the frame
N60 G1 X10 Y-10 G41 OFFN=1  
Square roughing; allowance 1mm
N70 X-10
N80 Y10
N90 X10
N100 Y-10
N110 G0 Z20 G40 OFFN=0  
Tool change
N120 T2 D1 X15 Y-15
N130 Z10 G41
N140 G1 X10 Y-10  
Square finishing
N150 X-10
N160 Y10
N170 X10
N180 Y-10
N190 Z20 G40  
Deselect frame
N200 TRANS
N210 TRAFOOF
N220 G0 X20 Z10 SPOS=45  
Approach initial position
N230 M30
7.3 Cylinder surface transformation: TRACYL

Programming

TRACYL(d) or TRACYL(d, t)

TRAFOOF

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACYL(d)</td>
<td>Activates the first TRACYL function specified in the channel machine data.</td>
</tr>
<tr>
<td></td>
<td>d is the parameter for the working diameter.</td>
</tr>
<tr>
<td>TRACYL(d, n)</td>
<td>Activates the n-th TRACYL function specified in the channel machine data.</td>
</tr>
<tr>
<td></td>
<td>The maximum for n is 2, TRACYL(d,1) corresponds to TRACYL(d).</td>
</tr>
<tr>
<td>d</td>
<td>Value for the working diameter. The working diameter is double the</td>
</tr>
<tr>
<td></td>
<td>distance between the tool tip and the turning center.</td>
</tr>
<tr>
<td>TRAFOOF</td>
<td>Transformation OFF (BCS and MCS are once again identical).</td>
</tr>
<tr>
<td>OFFN</td>
<td>Offset contour normal: Distance of the groove side from the programmed</td>
</tr>
<tr>
<td></td>
<td>reference contour</td>
</tr>
</tbody>
</table>

An active TRACYL transformation is also disabled if one of the remaining transformations is activated in the particular channel (e.g. TRANSMIT, TRAANG, TRAORI).

Function

Cylinder peripheral curve transformation

TRACYL

The TRACYL cylinder peripheral curve transformation facilitates the following performance:

Machining
- longitudinal grooves on cylindrical structures,
- transverse grooves on cylindrical structures,
- grooves in any direction on cylindrical structures.

The progression of the grooves is programmed in relation to the handled, even, peripheral surface of the cylinder.
There are two instances of cylinder surface coordinate transformation:

- without groove side offset (TRAFO_TYPE_n=512)
- with groove side offset (TRAFO_TYPE_n=513)

Without groove side offset:
The control transforms the programmed traversing movements of the cylinder coordinate system to the traversing movements of the real machine axes:
- Rotary axis
- Infeed axis perpendicular to the rotary axis
- Longitudinal axis parallel to the rotary axis.

The linear axes are positioned perpendicular to one another. The infeed axis cuts the rotary axis.

With groove side offset:
Kinematics as above, but in addition
- longitudinal axis parallel to the peripheral direction.

The linear axes are positioned perpendicular to one another.

The velocity control considers the limitations defined for rotary motion.
Groove cross section
In axis configuration 1, grooves alongside the rotary axis are only limited in parallel if the groove width corresponds exactly to the tool radius.

Grooves parallel to the circumference (transverse grooves) are not parallel at the start and at the end.

Offset contour normal OFFN (513)
To mill grooves with TRACYL, in
- the parts program the groove center line
- is programmed via OFFN half the width of the groove.

OFFN is only effective when tool radius compensation is selected, to avoid damaging the groove side. Furthermore, OFFN >= tool radius should also be the case to stop damage occurring to the opposite side of the groove.

A parts program for milling a groove generally comprises the following steps:
1. Select tool
2. Select TRACYL
3. Select suitable coordinate offset (frame)
4. Position
5. Program OFFN
6. Select TRC
7. Approach block (position TRC and approach groove side)
8. Groove center line contour
9. Deselect TRC
10. Retraction block (retract TRC and move away from groove side)
11. Position
12. TRAFOOF
13. Re-select original coordinate shift (frame)
Special features:

- **TRC selection:**
  TRC is not programmed in relation to the groove side, but relative to the programmed groove center line. To prevent the tool traveling to the left of the groove side, G42 is entered (instead of G41). You avoid this if in OFFN, the groove width is entered with a negative sign.

- **OFFN acts differently with TRACYL than it does without TRACYL.** As, even without TRACYL, OFFN is included when TRC is active, OFFN should be reset to zero after TRAFOOF.

- **It is possible to change OFFN within a parts program.** This could be used to shift the groove center line from the center (see diagram).

- **Guiding grooves:**
  TRACYL does not create the same groove for guiding grooves as it would be with a tool with the diameter producing the width of the groove. It is basically not possible to create the same groove side geometry with a smaller cylindrical tool as it is with a larger one. TRACYL minimizes the error. To avoid problems of accuracy, the tool radius should only be slightly smaller than half the groove width.

Note:

**OFFN and TRC**

- With TRAFO_TYPE_n = 512, the value acts under OFFN as an allowance for TRC.

- With TRAFO_TYPE_n = 513, half the groove width is programmed in OFFN. The contour is retracted with OFFN-TRC.
For cylinder peripheral curve transformation with groove side compensation, the axis used for compensation should be positioned at zero (y=0), so that the groove centric to the programmed groove center line is finished.

Rotary axis
The rotary axis cannot be programmed, as it is assigned by a geometry axis and is thus not directly programmable as a channel axis.

Axis utilization
The following axes cannot be used as a positioning axis or a reciprocating axis:
- the geometry axis in the peripheral direction of the cylinder peripheral surface (Y axis)
- the additional linear axis for groove side compensation (Z axis).

Tool definition
The following example is suitable for testing the parameterization of the TRACYL cylinder transformation:

<table>
<thead>
<tr>
<th>Tool parameters number (DP)</th>
<th>Meaning</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_DP1[1,1]=120$</td>
<td>Tool type</td>
<td>Milling cutter</td>
</tr>
<tr>
<td>$TC_DP2[1,1]=0$</td>
<td>Tool point direction</td>
<td>For turning tools only</td>
</tr>
<tr>
<td>Geometry</td>
<td>Tool length compensation</td>
<td></td>
</tr>
<tr>
<td>$TC_DP3[1,1]=8.$</td>
<td>Length compensation vector</td>
<td>Calculation depending</td>
</tr>
<tr>
<td>$TC_DP4[1,1]=9.$</td>
<td></td>
<td>on type and plane</td>
</tr>
<tr>
<td>$TC_DP5[1,1]=7.$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geometry</td>
<td>Radius</td>
<td></td>
</tr>
<tr>
<td>$TC_DP6[1,1]=6.$</td>
<td>Radius</td>
<td>Tool radius</td>
</tr>
<tr>
<td>$TC_DP7[1,1]=0$</td>
<td>Slot width b for slotting saw, rounding radius for milling tools</td>
<td></td>
</tr>
<tr>
<td>$TC_DP8[1,1]=0$</td>
<td>Overhang k</td>
<td>For slotting saw only</td>
</tr>
<tr>
<td>$TC_DP9[1,1]=0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP10[1,1]=0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP11[1,1]=0$</td>
<td>Angle for cone milling tools</td>
<td></td>
</tr>
<tr>
<td>Wear</td>
<td>Tool length and radius compensation</td>
<td></td>
</tr>
<tr>
<td>$TC_DP12[1,1]=0$</td>
<td>Remaining parameters to $TC_DP24=0</td>
<td>Base dimensions/adapter</td>
</tr>
</tbody>
</table>
7.3 Cylinder surface transformation: TRACYL

Programming example

N10 T1 D1 G54 G90 F5000 G94  
Tool selection, clamping compensation
N20 SPOS=0  
Approach initial position
N30 G0 X25 Y0 Z105 CC=200  
Enable cylinder peripheral curve transformation
N40 TRACYL (40)  
Plane selection
N50 G19

Making a hook-shaped groove:

N60 G1 X20  
Infeed tool to groove base
N70 OFFN=12  
Define 12mm groove side spacing relative to groove center line
N80 G1 Z100 G42  
Approach right side of groove
N90 G1 Z50  
Groove cut parallel to cylinder axis
N100 G1 Y10  
Groove cut parallel to circumference
N110 OFFN=4 G42  
Approach left side of the groove; define 4mm groove side spacing relative to the groove center line
N120 G1 Y70  
Groove cut parallel to circumference
N130 G1 Z100  
Groove cut parallel to cylinder axis
N140 G1 Z105 G40  
Return from groove side
N150 G1 X25  
Retraction
N160 TRAFOOF
N170 G0 X25 Y0 Z105 CC=200  
Approach initial position
N180 M30
7.4 Inclined axis: TRAANG

Programming

TRAANG (α) or TRAANG (α, n)
TRAFOOF

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAANG</td>
<td>If angle α is omitted or zero is entered, the transformation is activated with the parameterization of the previous selection. The default selection according to the machine data applies for the initial selection.</td>
</tr>
<tr>
<td>TRAANG (α)</td>
<td>Activates the first specified inclined axis transformation</td>
</tr>
<tr>
<td>TRAANG (α, n)</td>
<td>Activates the n-th specified transformation. Inclined axis. The maximum for n is 2. TRAANG(α,1) corresponds to TRAANG(α).</td>
</tr>
<tr>
<td>α</td>
<td>Angle of the inclined axis</td>
</tr>
<tr>
<td>TRAFOOF</td>
<td>Transformation OFF</td>
</tr>
</tbody>
</table>

If α (angle) is omitted or zero is entered, the transformation is activated with the parameterization of the previous selection. The default selection according to the machine data applies for the initial selection. (response up to SW < 6.4, for later versions, see below).

An active TRAANG transformation is also disabled if one of the remaining transformations is activated in the particular channel. (e.g. TRACYL, TRANSMIT, TRAORI).
If \( \alpha \) (angle) is omitted (e.g. TRAANG(), TRAANG(,n)), the transformation is activated with the parameterization of the previous selection. The default selection according to the machine data applies for the initial selection.

An angle \( \alpha = 0 \) (e.g. TRAANG(0), TRAANG(0,n)) is a valid parameter setting and no longer corresponds to omitting the parameter as it did in former versions.

Permissible values for \( \alpha \) are:

-90 degrees < \( \alpha \) < + 90 degrees

**Function**

The inclined axis function is intended for grinding technology and facilitates the following performance:

- Machining with an oblique infeed axis
- A cartesian coordinate system can be used for programming.
- The control transforms the programmed traversing movements of the cartesian coordinate system to the traversing movements of the real machine axes (default situation): inclined infeed axis.

The following machining operations are possible:

1. longitudinal grinding
2. face grinding
3. grinding a specific contour
4. oblique plunge-cut grinding
The following settings are defined in machine data:

- the angle between a machine axis and the oblique axis
- the position of the zero point of the tool relative to the origin of the coordinate system specified by the "inclined axis" function
- the velocity reserve held ready on the parallel axis for the compensating movement.
- the axis acceleration reserve held ready on the parallel axis for the compensating movement.

**Axis configuration**

To be able to program in the cartesian coordinate system, the control must be told the relationship between this coordinate system and the actually existing machine axes (MU, MZ):

- Geometry axes designation
  - Assignment of geometry axes to channel axes
    - general situation (inclined axis not active)
    - inclined axis active
  - Assignment of channel axes to machine axis numbers
- Spindle identification
- Machine axis name assignment.

Apart from "inclined axis active", the procedure corresponds to the procedure for normal axis configuration.
Programming example

```
N10 G0 G90 Z0 MU=10 G54 F5000 -> Tool selection, clamping compensation
-> G18 G64 T1 D1 Plane selection
N20 TRAANG(45) Enable inclined axis transformation
N30 G0 Z10 X5 Approach initial position
N40 WAITP(Z) Enable axis for reciprocation
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=-2 Reciprocation, until dimension reached
 -> OST2[Z]=-2 FA[Z]=5000 (for reciprocation, see chapter 9)
N60 OS[Z]=1
N70 POS[X]=4.5 FA[X]=50
N80 OS[Z]=0
N90 WAITP(Z) Enable reciprocating axes as positioning axes
N100 TRAFOOF Switch off transformation
N110 G0 Z10 MU=10 Retraction
N120 M30
```

--> program in a single block
7.4 Inclined axis: TRAANG

7.4.1 Inclined axis programming: G05, G07 (SW 5.3 and higher)

**Programming**

- G07
- G05

**Explanation of the commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G07</td>
<td>Approach starting position</td>
</tr>
<tr>
<td>G05</td>
<td>Activates oblique plunge-cutting</td>
</tr>
</tbody>
</table>

The commands G07/G05 are used to make it easier to program the inclined axes.

Positions can be programmed and displayed in the cartesian coordinate system. Tool compensation and zero offset are included in cartesian coordinates. After the angle for the inclined axis is programmed in the NC-program, the starting position can be approached (G07) and then the oblique plunge-cutting (G05) performed.

In Jog-mode, the movement of the grinding wheel can either be cartesian or in the direction of the inclined axis (the display stays cartesian). All that moves is the real U-axis, the Z-axis display is updated.

- In Jog-mode, repos-offsets must be returned using cartesian coordinates.
- In Jog-mode with active "PTP-travel", the cartesian operating range limit is monitored for overtravel and the relevant axis is braked beforehand. If "PTP-travel" is not active, the axis can be traversed right up to the operating range limit.

References: /FB2/ F2: 3-5-axis transformation, Chapter 2 "Cartesian PTP-travel".
Programming example

<table>
<thead>
<tr>
<th>N..</th>
<th>Program angle for inclined axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>N20 G07 X70 Z40 F4000</td>
<td>Approach starting position</td>
</tr>
<tr>
<td>N30 G05 X70 F100</td>
<td>Oblique plunge-cutting</td>
</tr>
<tr>
<td>N40 ...</td>
<td></td>
</tr>
</tbody>
</table>
7.5 Constraints when selecting a transformation

Transformations can be selected via a parts program or MDA. Please note the following:

- No intermediate movement block is inserted (chamfer/radii).
- Spline block sequences must be excluded; if not, a message is displayed.
- Fine tool compensation must be deselected (FTOCOF); if not a message is displayed.
- Tool radius compensation must be deselected (G40); if not a message is displayed.
- The control adopts an activated tool length compensation into the transformation.
- The control deselects the current frame active before the transformation.
- The control deselects an active operating range limit for axes affected by the transformation (corresponds to WALIMOF).
- Protection zone monitoring is deselected.
- Continuous-path mode and smoothing are interrupted.
- DRF offsets in the axes involved in the transformation must not change between processing in preprocessing and in main run (SW 3 and earlier).
- All the axes specified in the machine data must be synchronized relative to a block.
- Axes that are exchanged are exchanged back; if not, a message is displayed.
- A message is output for dependent axes.

**Tool change**

A tool change is only permissible if tool radius compensation is deselected.

A change of tool length compensation and a tool radius compensation selection/deselection must not be programmed in the same block.
Frame change
All instructions that only relate to the basic coordinate system are legal (frame, tool radius compensation). However, a frame change with G91 (incremental dimension) – unlike with an inactive transformation – is not handled separately. The increment to be traveled is evaluated in the workpiece coordinate system of the new frame – regardless of which frame was effective in the previous block.

Exclusions
Axes affected by the transformation cannot be used
• as the preset axis (alarm)
• for approaching a checkpoint (alarm)
• for referencing (alarm).
### 7.6 Deselect transformation: TRAFOOF

#### Programming

```plaintext
TRAFOOF
```

#### Explanation of the commands

| TRAFOOF | Disables all the active transformations/frames |

#### Function

The TRAFOOF command disables all the active transformations and frames.

Frames required after this must be activated by renewed programming.

Please note the following:
The same restrictions as for selection are applicable to deselecting the transformation (see previous section "Constraints when selecting a transformation").
7.7 Chained transformations

As from SW 5, two transformations can always be enabled in succession (chained), so that the motion components for the axes from the first transformation are the input data for the chained second transformation. The motion components from the second transformation act on the machine axes.

- In SW 5, the chain can consist of two transformations.
- The second transformation must be "inclined axis" (TRAANG).
- Possible first transformations include:
  - orientation transformations (TRAORI), incl. universal milling head
  - TRANSMIT
  - TRACYL
  - TRAANG.

Applications

- Grinding contours that have been programmed as the surface line of a cylinder development (TRACYL) with an obliquely positioned grinding wheel, e.g. tool grinding.
- Finishing a contour generated with TRANSMIT that is not round with an obliquely positioned grinding wheel.

It is a condition of using the activate command for a chained transformation that the individual transformations to be chained and the chained transformation to be activated are defined by the machine data. The constraints and special cases specified in the individual descriptions for the transformations must also be observed when they are used within a chain.
Additional notes

Information on configuring the machine data of the transformations can be found in the descriptions of the functions: M1 and F2.

Machine manufacturer (MH7.1)

Take note of information provided by the machine manufacturer on any transformations predefined by the machine data.

Transformations and chained transformations are options. The current catalog always provides information about the availability of specific transformations in the chain in specific controls.

The commands available for chained transformations are:

- TRACON to activate and
- TRAFOOF to deactivate.

Activate

Programming

TRACON(trf, par)

This activates a chained transformation.

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trf</td>
<td>The number of the chained transformation: 0 or 1 for the first/only chained transformation. If nothing is programmed in this position, this means the same as specifying the value 0 or 1, i.e. the first/only transformation is activated. 2 for the second chained transformation. (values not equal to 0–2 generate an error alarm).</td>
</tr>
</tbody>
</table>
one or more parameters separated by a comma for the transformations in the chain expecting parameters. For example, the angle of the inclined axis. If parameters are not set, the defaults or the parameters last used take effect. Commas must be used to ensure that the specified parameters are evaluated in the sequence in which they are expected, if defaults are to act for previous parameters. It is particularly important when specifying at least one parameter that this is preceded by a comma, even if it is not necessary to specify trf, thus for example TRACON( , 3.7).

**Function**

This activates the chained transformation. A previously activated other transformation is implicitly disabled by TRACON().

A tool is always assigned to the first transformation of a chain. The subsequent transformation then behaves as if the active tool length were zero. Only the base lengths of a tool (_BASE_TOOL_) set via machine data are active for the first transformation of the chain.

**Deactivate**

**Programming**

**TRAFOOF**

**Function**

The command deactivates the last active (chained) transformation.
7.8 Switchable geometry axes, GEOAX

Programming

GEOAX(n,channel axis,n,channel axis,...)

GEOAX()

Explanation of the parameters

| GEOAX(n,channel axis,n,channel axis,...) | Switch the geometry axes. |
| GEOAX() | Call the basic configuration of the geometry axes |

n
Number of the geometry axis (n=1, 2 or 3) to be assigned to another channel axis.
n=0: remove the specified channel axis from the geometry axis grouping without replacement.

Channel axis
Name of the channel axis to be accepted into the geometry axis grouping.

Function

The "switchable geometry axes" function allows the geometry axis grouping configured via machine data to be modified from the parts program. A channel axis defined as a synchronized special axis can replace any geometry axis.

Example:
A tool carriage can be traversed over channel axes X1, Y1, Z1, Z2. In the parts program, axes Z1 and Z2 should be used alternately as geometry axis Z. GEOAX is used in the parts program to switch between the axes.
After activation, the connection X1, Y1, Z1 is effective (adjustable via MD).

N100 GEOAX (3,Z2)
N110 G1 ....
N120 GEOAX (3,Z1)

Channel axis Z2 functions as the Z axis
Channel axis Z1 functions as the Z axis
Sequence

Geometry axis number
In the command GEOAX(n,channel axis...) the number n designates the geometry axis to which the subsequently specified channel axis is to be assigned.
Geometry axis numbers 1 to 3 (X, Y, Z axis) are permissible for loading a channel axis.
n = 0 removes an assigned channel axis from the geometry axis grouping without reassigning the geometry axis.

After the transition, an axis replaced by switching in the geometry axis grouping is programmable as a special axis via its channel name.

Switching over the geometry axes deletes all the frames, protection zones and operating range limits.

Polar coordinates:
As with a change of plane (G17–G19), replacing geometry axes with GEOAX sets the modal polar coordinates to the value 0.

DRF, ZO:
Any existing handwheel offset (DRF) or an external zero offset, will stay active after the switchover.

Exchange axis positions
It is also possible to change positions within the geometry axis grouping by reassigning the axis numbers to already assigned channel axes.

N... GEOAX (1, XX, 2, YY, 3, ZZ)
Channel axis XX is the first, YY the second and ZZ the third geometry axis,

N... GEOAX (1, U, 2, V, 3, W)
Channel axis U is the first, V the second and W the third geometry axis.
Prerequisites and restrictions

1. It is not possible to switch the geometry axes over during:
   - an active transformation,
   - an active spline interpolation,
   - an active tool radius compensation,
   - an active fine tool compensation

2. If the geometry axis and the channel axis have the same name, it is not possible to change the particular geometry axis.

3. None of the axes involved in the switchover can be involved in an action that might persist beyond the block limits, as is the case, for example, with positioning axes of type A or with following axes.

4. The GEOAX command can only be used to replace geometry axes that already existed at power ON (i.e. no newly defined ones).

5. Using GEOAX for axis replacement while preparing the **contour table** (CONTPRON, CONTDCON) produces an alarm.

Deactivating switchover

The command GEOAX() calls the basic configuration of the geometry axis grouping.

After POWER ON and when switching over to reference point approach mode, the basic configuration is reset automatically.

Additional notes

Transition and tool length compensation

An active tool length compensation is also effective after the transition. However, for the newly adopted or repositioned geometry axes, it counts as not retracted.
So accordingly, at the first motion command for these geometry axes, the resultant travel path comprises the sum of the tool length compensation and the programmed travel path.
Geometry axes that retain their position in the axis grouping during a switchover, also keep their status with regard to tool length compensation.

**Geometry axis configuration and transformation change**

The geometry axis configuration applicable in an active transformation (defined via the machine data) cannot be modified by using the "switchable geometry axes" function.

If it is necessary to change the geometry axis configuration in connection with transformations, this is only possible via an additional transformation.

A geometry axis configuration modified via GEOAX is deleted by activating a transformation.

If the machine data settings for the transformation and for switching over the geometry axes contradict one another, the settings in the transformation take precedence.

**Example:**

A transformation is active. According to the machine data, the transformation should be retained during a RESET, however, at the same time, a RESET should produce the basic configuration of the geometry axes. In this case, the geometry axis configuration is retained as specified by the transformation.
# Switchable geometry axes, GEOAX

A machine has six channel axes called XX, YY, ZZ, U, V, W. The basic setting of the geometry axis configuration via the machine data is:

- Channel axis XX = 1st geometry axis (X axis)
- Channel axis YY = 2nd geometry axis (Y axis)
- Channel axis ZZ = 3rd geometry axis (Z axis)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>GEOAX()</td>
<td>The basic configuration of the geometry axes is effective.</td>
</tr>
<tr>
<td>N20</td>
<td>G0 X0 Y0 Z0 U0 V0 W0</td>
<td>All the axes in rapid traverse to position 0.</td>
</tr>
<tr>
<td>N30</td>
<td>GEOAX(1,U,2,V,3,W)</td>
<td>Channel axis U becomes the first (X), V the second (Y), W the third geometry axis (Z).</td>
</tr>
<tr>
<td>N40</td>
<td>GEOAX(1,XX,3,ZZ)</td>
<td>Channel axis XX becomes the first (X), ZZ the third geometry axis (Z). Channel axis V stays as the second geometry axis (Y).</td>
</tr>
<tr>
<td>N50</td>
<td>G17 G2 X20 I10 F1000</td>
<td>Full circle in the X, Y plane. Channel axes XX and V traverse.</td>
</tr>
<tr>
<td>N60</td>
<td>GEOAX(2,W)</td>
<td>Channel axis W becomes the second geometry axis (Y).</td>
</tr>
<tr>
<td>N80</td>
<td>G17 G2 X20 I10 F1000</td>
<td>Full circle in the X, Y plane. Channel axes XX and W traverse.</td>
</tr>
<tr>
<td>N90</td>
<td>GEOAX()</td>
<td>Reset to initial state.</td>
</tr>
<tr>
<td>N100</td>
<td>GEOAX(1,U,2,V,3,W)</td>
<td>Channel axis U becomes the first (X), V the second (Y), W the third geometry axis (Z).</td>
</tr>
<tr>
<td>N110</td>
<td>G1 X10 Y10 Z10 XX=25</td>
<td>Channel axes U, V, W each traverse to position 10, XX as the special axis traverses to position 25.</td>
</tr>
<tr>
<td>N120</td>
<td>GEOAX(0,V)</td>
<td>V is removed from the geometry axis grouping. U and W are still the first (X) and third geometry axis (Z). The second geometry axis (Y) remains unassigned.</td>
</tr>
<tr>
<td>N130</td>
<td>GEOAX(1,U,2,V,3,W)</td>
<td>Channel axis U stays the first (X), V becomes the second (Y), W stays the third geometry axis (Z).</td>
</tr>
<tr>
<td>N140</td>
<td>GEOAX(3,V)</td>
<td>V becomes the third geometry axis (Z), which overwrites W and thus removes it from the geometry axis grouping. The second geometry axis (Y) is still unassigned.</td>
</tr>
</tbody>
</table>
Tool Offsets

8.1 Offset memory ........................................................................................................ 8-314
8.2 Language commands for tool management ........................................................... 8-316
8.3 Online tool offset PUTFTOCF, PUTFTOC, FTOCON, FTOCOF ......................... 8-319
8.4 Maintain tool radius compensation at constant level, CUTCONON
   (SW 4 and higher) ............................................................................................. 8-325
8.5 Activate 3D tool offsets ....................................................................................... 8-328
8.6 Tool orientation .................................................................................................... 8-336
8.7 Free assignment of D numbers, cutting edge number CE (SW 5 and higher) .... 8-341
  8.7.1 Check D numbers (CHKDNO) ................................................................. 8-342
  8.7.2 Renaming D numbers (GETDNO, SETDNO) ............................................... 8-343
  8.7.3 T numbers for the specified D number (GETACTTD) ................................. 8-344
  8.7.4 Set final D numbers to invalid ................................................................. 8-345
8.8 Toolholder kinematics ....................................................................................... 8-346
8.1 Offset memory

Structure of the offset memory
Every data field can be invoked with a T and D number (except "Flat D No."); in addition to the geometrical data for the tool, it contains other information such as the tool type.

SW 4 and higher
The "Flat D No. structure" is used if tool management takes place outside the NCK. In this case, the D numbers are generated with the associated tool offset blocks without being assigned to tools. You can still program in the parts program using T. However, this T does not relate to the programmed D number.

Several entries exist for the geometric variables (e.g. length 1 or radius). These are added together to produce a value (e.g. total length 1, total radius) which is then used for the calculations.

Offset values not required must be assigned the value zero.

The individual values of the offset memories P1 to P25 can be read from and written to the program via system variable.
### Tool offsets

#### Offset memory

<table>
<thead>
<tr>
<th>Tool parameters</th>
<th>Meaning</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number (DP)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{DP} \ 1$</td>
<td>Tool type</td>
<td>For overview see list</td>
</tr>
<tr>
<td>$TC_{DP} \ 2$</td>
<td>Tool point direction</td>
<td>For turning tools only</td>
</tr>
</tbody>
</table>

#### Geometry

<table>
<thead>
<tr>
<th>Tool length compensation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 3$</td>
</tr>
<tr>
<td>$TC_{DP} \ 4$</td>
</tr>
<tr>
<td>$TC_{DP} \ 5$</td>
</tr>
</tbody>
</table>

#### Geometry

<table>
<thead>
<tr>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 6$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slot width b for slotting saw, rounding radius for milling tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 7$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overhang k</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 8$</td>
</tr>
</tbody>
</table>

#### Wear

<table>
<thead>
<tr>
<th>Tool length and radius compensation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 12$</td>
</tr>
<tr>
<td>$TC_{DP} \ 13$</td>
</tr>
<tr>
<td>$TC_{DP} \ 14$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 15$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slot width b for slotting saw, rounding radius for milling tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 16$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overhang k</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 17$</td>
</tr>
</tbody>
</table>

#### Base dimensions/adapter

<table>
<thead>
<tr>
<th>Tool length compensation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 21$</td>
</tr>
<tr>
<td>$TC_{DP} \ 22$</td>
</tr>
<tr>
<td>$TC_{DP} \ 23$</td>
</tr>
</tbody>
</table>

#### Technology

<table>
<thead>
<tr>
<th>Clearance angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_{DP} \ 24$</td>
</tr>
</tbody>
</table>

### Additional notes

All other parameters are reserved.

**Machine manufacturer**

User cutting edge data can be configured via MD.
8.2 Language commands for tool management

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=&quot;WZ&quot;</td>
<td>Select tool with name</td>
</tr>
<tr>
<td>NEWT(&quot;WZ&quot;, DUPLO_NO)</td>
<td>Create new tool, duplo number optional</td>
</tr>
<tr>
<td>DELT(&quot;WZ&quot;, DUPLO_NO)</td>
<td>Delete tool, duplo number optional</td>
</tr>
<tr>
<td>GETT(&quot;WZ&quot;, DUPLO_NO)</td>
<td>Determine T number</td>
</tr>
<tr>
<td>SETPIECE(x,y)</td>
<td>Set piece number</td>
</tr>
<tr>
<td>GETSELT(x)</td>
<td>Read preselected tool number (T No.)</td>
</tr>
<tr>
<td>&quot;WZ&quot;</td>
<td>Tool name</td>
</tr>
<tr>
<td>DUPLO_NO</td>
<td>Quantity</td>
</tr>
<tr>
<td>x</td>
<td>Spindle number, entry optional</td>
</tr>
</tbody>
</table>

If you use the tool manager you can create and call tools by name, e.g. T="DRILL" or T="123".

NEWT function

With the NEWT function you can create a new tool with name in the NC program. The function automatically returns the T number created, which can subsequently be used to address the tool.

Return parameter=NEWT("WZ", DUPLO_NO)

If no duplo number is specified, this is generated automatically by the tool manager.

Example:

```
DEF INT DUPLO_NO
DEF INT T_NO
DUPLO_NO = 7
T_NO=NEWT("DRILL", DUPLO_NO) Create new tool "DRILL" with duplo number 7. The T number created is stored in T_NO.
```

DELT function

The DELT function can be used to delete a tool without referring to the T number.

DELT("WZ", DUPLO_NO)
**GETT function**

The GETT function returns the T number required to set the tool data for a tool known only by its name.

Return parameter = GETT("WZ", DUPLO_NO)

If several tools with the specified name exist, the T number of the first possible tool is returned.

Return parameter = –1: The tool name or duplo number cannot be assigned to a tool.

**Examples:**

T="DRILL"

R10=GETT("DRILL", DUPLO_NO)  \hspace{1cm} \text{Return T number for DRILL with duplo number = DUPLO_NO}

The "DRILL" must first be declared with NEWT or 

\$TC_TP1[ ].

\$TC_DP1[GETT("DRILL", DUPLO_NO),1]=100  \hspace{1cm} \text{Write a tool parameter with tool name}

**SETPIECE function**

This function is used to update the piece number monitoring data.

The function counts all of the tool edges which have been changed since the last activation of SETPIECE for the stated spindle number.

**SETPIECE(x,y)**

\begin{itemize}
  \item \textbf{\textit{x}}  \hspace{1cm} \text{Number of completed workpieces}
  \item \textbf{\textit{y}}  \hspace{1cm} \text{y spindle number, 0 stands for master spindle (default setting)}
\end{itemize}
**GETSELT function**

This function returns the T number of the tool preselected for the spindle. This function allows access to the tool offset data before M6 and thus establishes main run synchronization slightly earlier.

**Example for tool change with tool management**

**T1**  
Preselect tool, i.e. the tool magazine can be brought into the tool position parallel to machining.

**M6**  
Load preselected tool (depending on the setting in the machine data you can also program without M6).

**Example:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Preselect tool</td>
</tr>
<tr>
<td>M6</td>
<td>Load preselected tool</td>
</tr>
<tr>
<td>D1</td>
<td>Select tool length compensation</td>
</tr>
<tr>
<td>G1 X10</td>
<td>Machining with T1</td>
</tr>
<tr>
<td>T=&quot;DRILL&quot;</td>
<td>Preselect drill</td>
</tr>
<tr>
<td>D2 Y20</td>
<td>Change cutting edge T1</td>
</tr>
<tr>
<td>X10</td>
<td>Machining with T1</td>
</tr>
<tr>
<td>M6</td>
<td>Load tool drill</td>
</tr>
<tr>
<td>SETPIECE(4)</td>
<td>Number of completed workpieces</td>
</tr>
<tr>
<td>D1 G1 X10</td>
<td>Machining with drill</td>
</tr>
</tbody>
</table>

A complete list of all variables required for tool management is given in the list of system variables in the Appendix.
8.3 Online tool offset PUTFTOCF, PUTFTOC, FTOCON, FTOCOF

Programming:

- **FCTDEF**(Polynomial no., LLimit, ULimit, $a_0$, $a_1$, $a_2$, $a_3$)
- **PUTFTOCF**(Polynomial No., Ref_value, Length1_2_3, Channel, Spindle)
- **PUTFTOC**(Value, Length1_2_3, Channel, Spindle)
- **FTOCON**
- **FTOCOF**

Explanation of the commands

- **PUTFTOCF**: Write online tool offsets continuously
- **FCTDEF**: Define parameters for PUTFTOCF function
- **PUTFTOC**: Write online tool offsets discretely
- **FTOCON**: Activate online tool offsets
- **FTOCOF**: Deactivate online tool offsets

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial_No.</td>
<td>Values 1-3: A maximum of three polynomials can be programmed at the same time; polynomials up to 3rd degree</td>
</tr>
<tr>
<td>Ref_value</td>
<td>Reference value from which the offset is derived</td>
</tr>
<tr>
<td>Length1_2_3</td>
<td>Wear parameter into which the tool offset value is added</td>
</tr>
<tr>
<td>Channel</td>
<td>Number of channel in which the tool offset is activated; specified only if the channel is different to the present one</td>
</tr>
<tr>
<td>Spindle</td>
<td>Number of the spindle on which the online tool offset acts; only needs to be specified for inactive grinding wheels</td>
</tr>
<tr>
<td>LLimit</td>
<td>Lower limit</td>
</tr>
<tr>
<td>ULimit</td>
<td>Upper limit</td>
</tr>
<tr>
<td>$a_0$, $a_1$, $a_2$, $a_3$</td>
<td>Coefficients of polynomial function</td>
</tr>
<tr>
<td>Value</td>
<td>Value added in the wear parameter</td>
</tr>
</tbody>
</table>
Function

The function makes immediate allowance for tool offsets resulting from machining by means of online tool length compensation (e.g. CD dressing: The grinding wheel is dressed parallel to machining). The tool length compensation can be changed from the machining channel or a parallel channel (dresser channel).

Online tool offset can be applied only to grinding tools.

General information about online TO

Depending on the timing of the dressing process, the following functions are used to write the online tool offsets:

- Continuous write, non-modal: PUTFTOCF
- Continuous write, modally: ID=1 DO FTOC (see section synchronized actions)
- Discrete write: PUTFTOC

In the case of a continuous write (for each interpolation pulse) following activation of the evaluation function each change is calculated additively in the wear memory in order to prevent setpoint jumps.

In both cases:

- The online tool offset can act on each spindle and lengths 1, 2 or 3 of the wear parameters.
- The assignment of the lengths to the geometry axes is made with reference to the current plane.
- The assignment of the spindle to the tool is made with reference to the tool data with GWPSON or TMON as long as it is not the active grinding wheel (see Programming Guide "Fundamentals").
An offset is always applied for the wear parameters for the current tool side or for the left-hand tool side on inactive tools.

Where the offset is identical for several tool sides, the values should be transferred automatically to the second tool side by means of a chaining rule (see Operator's Guide for description).

If online offsets are defined for a machining channel, you cannot change the wear values for the current tool on this channel from the machining program or by means of an operator action.

The online tool offset is also applied with respect to the constant grinding wheel peripheral speed (GWPS) in addition to tool monitoring (TMON) and centerless grinding (CLGON).

**Sequence**

**PUTFTOCF = Continuous write**

The dressing process is performed at the same time as machining:
Dress across complete grinding wheel width with dresser roll or dresser diamond from one side of a grinding wheel to the other.

Machining and dressing can be performed on different channels. If no channel is programmed, the offset takes effect in the active channel.

PUTFTOCF(Polynomial_No., Ref_value, Length1_2_3, Channel, Spindle)

Tool offset is changed continuously on the machining channel according to a polynomial function of the first, second or third degree, which must have been defined previously with FCTDEF.
The offset, e.g. changing actual value, is derived from the "Reference value" variable. If a spindle number is not programmed, the offset applies to the active tool.
Set parameters for FCTDEF function
The parameters are defined in a separate block:

FCTDEF(Polynomial_NO., LLimit, ULimit, a0, a1, a2, a3)

The polynomial can be a 1st, 2nd or 3rd degree polynomial.
The limit identifies the limit values (LLimit = lower limit, ULimit = upper limit).
Example:
Straight line (y = a0 + a1x) with gradient 1
FCTDEF(1, -1000, 1000, -$AA_IW[X], 1)

Write online offset discretely: PUTFTOC
This command can be used to write an offset value once. The offset is activated immediately on the target channel.
Application of PUTFTOC:
The grinding wheel is dressed from a parallel channel, but not at the same time as machining.

PUTFTOC(Value, Length1_2_3, Channel, Spindle)

The online tool offset for the specified length 1, 2 or 3 is changed by the specified value, i.e. the value is added to the wear parameter.

Include online tool offset: FTOCON, FTOCOF
The target channel can only receive online tool offsets when FTOCON is active.

- FTOCON must be written in the channel on which the offset is to be activated.
  With FTOCOF, the offset is no longer applied, however the complete value written with PUTFTOC is corrected in the tool edge-specific offset data.
- FTOCOF is always the reset setting.
- PUTFTOCF always acts on the subsequent traversing block.
- The online tool offset can also be selected modally with FTOC. Please refer to Section "Motion-synchronized actions" for more information.
Programming example

Task
On a surface grinding machine with the following parameters, the grinding wheel is to be dressed by the amount 0.05 after the start of the grinding movement at X100. The dressing amount is to be active with write online offset continuously.

Y: Infeed axis for the grinding wheel
V: Infeed axis for the dresser roll

Machine: Channel 1 with axes X, Z, Y
Dress: Channel 2 with axis V

Machining program in channel 1:

```plaintext
%_N_MACH_MPF
...
N110 G1 G18 F10 G90 Basic position
N120 T1 D1 Select current tool
N130 S100 M3 X100 Spindle on, travel to starting position
N140 INIT (2, "DRESS", "S") Select dressing program on channel 2
N150 START (2) Start dressing program on channel 2
N160 X200 Travel to destination position
N170 FTOCON Activate online offset
N... G1 X100 Continue machining
N... M30
```

Dressing program in channel 2:

```plaintext
%_N_DRESS_MPF
...
N40 FCTDEF (1, -1000, 1000, -$AA_IW[V], 1) Define function: Straight line
N50 PUTFTOCF (1, $AA_IW[V], 3, 1) Write online offset continuously: Length 3 of the current grinding wheel is derived from the movement of the V axis and corrected in channel 1.
N60 V-0.05 G1 F0.01 G91 Infeed movement for dressing, PUTFTOCF is only effective in this block
...
N... M30
```
Dressing program, modal:

```plaintext
%_N_DRESS_MPF
FCTDEF(1, -1000, 1000, -$AA_IW[V], 1)  
Define function:
ID=1 DO  FTOC(1, $AA_IW[V], 3, 1)  
Select online tool offset:
    Actual value of the V axis is the input value for polynomial 1; the result is added length 3 of the active grinding wheel in channel 1 as the offset value.
WAITM(1, 1, 2)  
Synchronization with machining channel
G1 V-0.05 F0.01, G91  
Infeed movement for dressing
G1 V-0.05 F0.02
...
CANCEL(1)  
Deselect online offset
...
```
8.4 Maintain tool radius compensation at constant level, CUTCONON (SW 4 and higher)

Programming:

CUTCONON
CUTCONOF

Explanation

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUTCONON</td>
<td>Activate the tool radius compensation constant function</td>
</tr>
<tr>
<td>CUTCONOF</td>
<td>Deactivate the constant function (default setting)</td>
</tr>
</tbody>
</table>

Function

The "tool radius compensation constant" function is used to suppress the tool radius compensation for a number of blocks while retaining the difference between the programmed and actual path of the tool center point accumulated in previous blocks as an offset.

This can be practical, for example, if several motion blocks are required at the reversal points during line-by-line milling but the contours (bypass strategies) generated by the tool radius compensation are not desirable.

It can be used according to the type of tool radius compensation (2½D, 3D face milling, 3D circumferential milling).

Sequence

Tool radius compensation is normally active before the compensation suppression and is still active when the compensation suppression is deactivated again.

The offset point at the end of block position is approached in the last motion block before CUTCONON.

All following blocks in which the compensation suppression is active are executed without compensation.
They are displaced, however, by the vector from the end point of the last compensation block to its offset point. The interpolation type of these blocks (linear, circular, polynomial) is arbitrary. The deactivation block of the compensation suppression, i.e. the block containing CUTCONOF, is usually corrected; it begins at the offset point of the start point. A linear block is inserted between this point and the end point of the previous block, i.e. the last programmed motion block with active CUTCONON. Circle blocks in which the circle plane is perpendicular to the compensation plane (vertical circles) are treated as if CUTCONON had been programmed in the blocks. This implicit activation of compensation suppression is automatically canceled in the first motion block which is not a circle of this type but which contains a traversing movement in the compensation plane. Vertical circles of this type can only occur with circumferential milling.

**Example**

```
N10 ;   ; Definition of tool d1
N20 $TC_DP1[1,1]= 110 ;   ; Type
N30 $TC_DP6[1,1]= 10. ;   ; Radius
N40
N50 X0 Y0 Z0 G1 G17 T1 D1 F10000
N60
N70 X20 G42 NORM
N80 X30
N90 Y20
N100 X10 CUTCONON;   ; Activate compensation suppression
N110 Y30 CONT ;   ; Insert bypass circle if necessary on deactivation of contour suppression
N120 X-10 CUTCONOF
N130 Y20 NORM ;   ; No bypass circle on deactivation of TRC
N140 X0 Y0 G40
N150 M30
```
### Additional notes

1. **CUTCONON** has no effect if tool radius compensation is not active (G40). An alarm is output. The G code remains active, however. This is significant if tool radius compensation is to be activated in a subsequent block with G41 or G42.

2. It is possible to change the G code in the 7th G code group (tool radius compensation; G40 / G41 / G42) when CUTCNONON is active. A change to G40 is effective immediately. The offset with which the previous blocks were traversed is applied.

3. If CUTCNONON or CUTCNONOF is programmed in a block without a traversing movement in the active compensation plane, the change does not become effective until the next block with such a traversing movement.

Further information: /FB/, W1 Tool Offset
8.5 Activate 3D tool offsets

Explanation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUT3DC</td>
<td>Activation of 3D radius offset for circumferential milling</td>
</tr>
<tr>
<td>CUT3DFS</td>
<td>3D tool offset for face milling with constant orientation. The tool orientation is determined by G17-G19 and is not influenced by Frames.</td>
</tr>
<tr>
<td>CUT3DFF</td>
<td>3D tool offset for face milling with constant orientation. The tool orientation is the direction determined by G17-G19 and possibly turned by a Frame.</td>
</tr>
<tr>
<td>CUT3DF</td>
<td>3D tool offset for face milling with orientation change (only with active 5-axes transformation).</td>
</tr>
<tr>
<td>G40 X Y Z</td>
<td>To deactivate: Linear block G0/G1 with geometry axes</td>
</tr>
<tr>
<td>ISD=Value</td>
<td>Insertion depth</td>
</tr>
</tbody>
</table>

The commands are modal and are in the same group as CUT2D and CUT2DF.

The command is not deselected until the next movement in the current plane is performed. This always applies to G40 and is independent of the CUT command.

Function

Tool orientation change is taken into account in tool radius compensation for cylindrical tools.

The same programming commands apply to 3D tool radius compensation as to 2D tool radius compensation. With G41/G42, the left/right-hand compensation is specified in the direction of movement. The approach method is always NORM.
8.5 Activate 3D tool offsets

Example

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>A0 B0 X0 Y0 Z0 F5000</td>
<td>Tool call, call tool offset values</td>
</tr>
<tr>
<td>N20</td>
<td>T1 D1</td>
<td>Transformation selection</td>
</tr>
<tr>
<td>N30</td>
<td>TRAORI(1)</td>
<td>3D tool radius compensation selection</td>
</tr>
<tr>
<td>N40</td>
<td>CUT3DC</td>
<td>Tool radius compensation selection</td>
</tr>
<tr>
<td>N50</td>
<td>G42 X10 Y10</td>
<td></td>
</tr>
<tr>
<td>N60</td>
<td>X60</td>
<td></td>
</tr>
<tr>
<td>N70</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Additional notes

Intermediate blocks are permitted with 3D tool radius compensation. The rules for 2 ½ D tool radius compensation apply.

3D tool radius compensation is only active when five-axis transformation is selected.

A circle block is always inserted at outside corners. G450/G451 have no effect.

The DISC command is ignored.
8.5 Activate 3D tool offsets

Difference between 2 ½ D and 3D tool radius compensation
In 3D tool radius compensation tool orientation can be changed.

2 ½ D tool radius compensation assumes the use of a tool with constant orientation.

3D tool radius compensation is also called 5D tool radius compensation, because in this case 5 degrees of freedom are available for the orientation of the tool in space.

Circumferential milling
The type of milling used here is implemented by defining a path (guide line) and the corresponding orientation. In this type of machining, the shape of the tool on the path is not relevant. The only deciding factor is the radius at the tool insertion point.

The 3D TRC function is limited to cylindrical tools.
Face milling
For this type of 3D milling, you require line-by-line definition of 3D paths on the workpiece surface. The tool shape and dimensions are taken into account in the calculations that are normally performed in CAM.
In addition to the NC blocks, the postprocessor writes the tool orientations (when five-axis transformation is active) and the G code for the desired 3D tool offset into the parts program.

This feature offers the machine operator the option of using slightly smaller tools than that used to calculate the NC paths.

Example:
NC blocks have been calculated with a 10mm mill. In this case, the workpiece could also be machined with a mill diameter of 9.9mm, although this would result in a different surface profile.
Mill shapes, tool data
The table below gives an overview of the tool shapes which may be used in face milling operations as well as tool data limit values.
The shape of the tool shaft is not taken into consideration – the tools 120 and 155 are identical in their effect.
If a different type number is used in the NC program than the one listed in the table, the system automatically uses tool type 110 die-sinking cutter. An alarm is output if the tool data limit values are violated.

<table>
<thead>
<tr>
<th>Milling tool type</th>
<th>Type No.</th>
<th>R</th>
<th>r</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cylindrical miller</td>
<td>110</td>
<td>&gt;0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Ball end mill</td>
<td>111</td>
<td>&gt;0</td>
<td>&gt;R</td>
<td>X</td>
</tr>
<tr>
<td>End mill, angle head cutter</td>
<td>120, 130</td>
<td>&gt;0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>End mill, angle head cutter with corner rounding</td>
<td>121, 131</td>
<td>&gt;r</td>
<td>&gt;0</td>
<td>X</td>
</tr>
<tr>
<td>Truncated cone mill</td>
<td>155</td>
<td>&gt;0</td>
<td>X</td>
<td>&gt;0</td>
</tr>
</tbody>
</table>

X= is not evaluated

Tool length compensation
The tool tip is the reference point for length compensation (intersection longitudinal axis/surface).
3D tool offset, tool change
A new tool with changed dimensions (R, r, a) or a different shape may be specified only through programming G41 or G42 (transition G40 to G41 or G42, reprogramming of G41 of G42).
This rule does not apply to any other tool data, e.g. tool lengths, so that tools to which such data apply can be fitted without reprogramming G41 or G42.

Correction of the path
With respect to face milling, it is advisable to examine what happens when the contact point "jumps" on the tool surface as shown in the example on the right where a convex surface is being machined with a vertically positioned tool.

As a general rule, it is advisable to select a tool shape and tool orientation that are suitable for producing the required surface profile.

The application shown in the example should therefore be regarded as a borderline case.

This borderline case is monitored by the control that detects abrupt changes in the machining point on the basis of angular approach motions between the tool and normal surface vectors. The control inserts linear blocks at these positions so that the motion can be executed.

These linear blocks are calculated on the basis of permissible angular ranges for the side angle stored in the machine data.

The system outputs an alarm if the limit values stored in the machine data are violated.
Path curvature
Path curvature is not monitored. In such cases, it is also advisable to use only tools of a type that do not violate the contour.

Insertion depth (ISD)
Program command ISD (insertion depth) is used to program the tool insertion depth for peripheral milling operations. This makes it possible to change the position of the machining point on the outer surface of the tool.

ISD specifies the distance between the cutter tip (FS) and the cutter reference point (FH). The point FH is produced by projecting the programmed machining point along the tool axis. ISD is only evaluated when 3D tool radius compensation is active.

Inside corners/outside corners
Inside and outside corners are handled separately. The term inside or outside corner depends on the tool orientation.

When changes occur in the orientation at a corner, the corner type can change during machining. If this happens, machining stops and an error message is generated.
Intersection procedure for 3D compensation (SW 5 and higher)

With 3D circumferential milling, G code G450/G451 is now evaluated at the outside corners; this means that the intersection of the offset curves can be approached. With SW 4 a circle was always inserted at the outside corners.

The new functionality is particularly advantageous for typical CAD-generated 3D programs. They often consist of short straight blocks (to approximate smooth curves), where the transitions are almost tangential between adjacent blocks.

Up to now, with tool radius compensation on the outside of the contour, circles were generally inserted to circumnavigate the outside corners. These blocks can be very short with almost tangential transitions, resulting in undesired drops in velocity.

In these cases, as with 2½ D radius compensation, both of the curves involved are lengthened and the intersection of both lengthened curves is approached.

The intersection is determined by extending the offset curves of both blocks and defining their intersection at the corner in the plane perpendicular to the tool orientation. If there is no such intersection, the corner is handled as previously, that is, a circle is inserted.

For more information about intersection procedure, see /FB/ W5, 3D Tool Radius Compensation
8.6 Tool orientation

Tool orientation is the term given to the geometrical alignment of the tool in space.

On a 5-axis machine tool, the tool orientation can be controlled with program commands.

Programming tool orientation

A change in tool orientation can be programmed by:

- Direct programming of the rotary axes
- Euler or RPY angle
- Direction vector
- LEAD/TILT (face milling)

The reference coordinate system is either the machine coordinate system (ORIMCS) or the current workpiece coordinate system (ORIWCS).

A change in orientation can be controlled by the following:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIC</td>
<td>Orientation and path movement in parallel</td>
</tr>
<tr>
<td>ORID</td>
<td>Orientation and path movement consecutively</td>
</tr>
<tr>
<td>OSOF</td>
<td>No orientation smoothing</td>
</tr>
<tr>
<td>OSC</td>
<td>Orientation constantly</td>
</tr>
<tr>
<td>OSS</td>
<td>Orientation smoothing only at beginning of block</td>
</tr>
<tr>
<td>OSSE</td>
<td>Orientation smoothing at beginning and end of block</td>
</tr>
<tr>
<td>ORIS</td>
<td>Speed of orientation change with active orientation smoothing in degrees per mm; valid for OSS and OSSE</td>
</tr>
</tbody>
</table>
Behavior at outside corners

A circle block with the radius of the cutter is always inserted at an outside corner.

The program commands ORIC and ORID can be used to define whether changes in orientation programmed between blocks N1 and N2 are executed before the beginning of the inserted circle block or at the same time.

If an orientation change is required at outside corners, this can be performed either at the same time as interpolation or separately together with the path movement.

With ORID, the inserted blocks are executed initially without a path movement. The circle block generating the corner is inserted immediately before the second of the two traversing blocks.

If several orientation blocks are inserted at an external corner and ORIC is selected, the circular movement is divided among the individual inserted blocks according to the values of the orientation changes.
Programming example for ORIC

If two or more blocks with orientation changes (e.g. \( A2= B2= C2= \)) are programmed between traversing blocks N10 and N20 and ORIC is active, the inserted circle block is divided among these intermediate blocks according to the values of the angle changes.

ORIC

<table>
<thead>
<tr>
<th>N8</th>
<th>A2=... B2=... C2=...</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>X... Y... Z...</td>
</tr>
<tr>
<td>N12</td>
<td>C2=... B2=...</td>
</tr>
<tr>
<td>N14</td>
<td>C2=... B2=...</td>
</tr>
</tbody>
</table>

The circle block inserted at the external corner is divided among N12 and N14 in accordance with the change in orientation. The circular movement and the orientation change are executed in parallel.

<table>
<thead>
<tr>
<th>N20</th>
<th>X =...Y=... Z=...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G1 F200</td>
</tr>
</tbody>
</table>
Programming example for ORID

If ORID is active, all the blocks between the two traversing blocks are executed at the end of the first traversing block. The circle block with constant orientation is executed immediately before the second traversing block.

<table>
<thead>
<tr>
<th>ORID</th>
<th>Blocks N12 and N14 are executed at the end of N10. The circle block with the current orientation is subsequently traversed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>N8</td>
<td>A2=... B2=... C2=...</td>
</tr>
<tr>
<td>N10</td>
<td>X... Y... Z...</td>
</tr>
<tr>
<td>N12</td>
<td>A2=... B2=... C2=...</td>
</tr>
<tr>
<td>N14</td>
<td>M20</td>
</tr>
<tr>
<td>N20</td>
<td>X... Y... Z...</td>
</tr>
</tbody>
</table>

The program command which is active in the first traversing block of an external corner determines the type of orientation change.
Without orientation change
If the orientation is not changed at the block boundary, the cross-section of the tool is a circle which touches both of the contours.

Programming example
Change the orientation at an internal corner

```plaintext
ORIC
N10 X ... Y ... Z ... G1 F500
N12 X ... Y ... Z ... A2=... B2=... C2=...
N15 X Y Z A2 B2 C2
```
8.7 Free assignment of D numbers, cutting edge number CE
(SW 5 and higher)

As of SW 5, you can use the D numbers as contour numbers. You can also address the number of the cutting edge via the address CE. You can use the system parameter $TC_{DPCE}$ to describe the cutting edge number.

Preset: Offset No. == Cutting edge No.
References: FB, W1 (Tool Offset)

Machine manufacturer (MH 8.12)

The maximum number of D numbers (cutting edge numbers) and maximum number of cutting edges per tool are defined via the machine data. The following commands only make sense when the maximum number of cutting edges (MD 18105) is greater than the number of cutting edges per tool (MD 18106). Please refer to the data of the machine tool manufacturer.

Additional notes

Besides the relative D number, you can also assign D numbers as 'flat' or 'absolute' D numbers (1–32000) without assigning a reference to a T number (inside the function 'flat D number structure').
8.7 Free assignment of D numbers, cutting edge number CE

8.7.1 Check D numbers (CHKDNO)

Programming:

\[
\text{state} = \text{CHKDNO}(\text{Tno1}, \text{Tno2}, \text{Dno})
\]

Explanation of the parameters

<table>
<thead>
<tr>
<th>state</th>
<th>TRUE:</th>
<th>The D numbers are assigned uniquely to the checked areas.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE:</td>
<td>There was a D number collision or the parameters are invalid. Tno1, Tno2 and Dno return the parameters that caused the collision. These data can now be evaluated in the parts program.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHKDNO(Tno1,Tno2)</th>
<th>All D numbers of the part specified are checked.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKDNO(Tno1)</td>
<td>All D numbers of Tno1 are checked against all other tools.</td>
</tr>
<tr>
<td>CHKKDN0</td>
<td>All D numbers of all tools are checked against all other tools.</td>
</tr>
</tbody>
</table>

Function

CHKDNO checks whether the available D numbers assigned are unique.
The D numbers of all tools defined in a TO unit must only be present once. Replacement tools are not considered.
8.7 Free assignment of D numbers, cutting edge number CE

8.7.2 Renaming D numbers (GETDNO, SETDNO)

Programming:

\[
d = \text{GETDNO}(t, \text{ce})
\]

\[
\text{state} = \text{SETDNO}(t, \text{ce}, d)
\]

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d)</td>
<td>D number of the cutting edge of the tool</td>
</tr>
<tr>
<td>(t)</td>
<td>T number of the tool</td>
</tr>
<tr>
<td>(\text{ce})</td>
<td>Cutting edge number (CE number) of the tool</td>
</tr>
<tr>
<td>\text{state}</td>
<td>Indicates whether the command could be executed (TRUE or FALSE).</td>
</tr>
</tbody>
</table>

Function

GETDNO
This command returns the D number of a particular cutting edge (\(\text{ce}\)) of a tool with tool number \(t\).
If there is no D number for the specified parameters, \(d\) is set to 0. If the D number is invalid, a value greater than 32000 is returned.

SETDNO
This commands assigns the value \(d\) of the D number to a cutting edge \(\text{ce}\) of tool \(t\). The result of this statement is returned via state (TRUE or FALSE).
If there is no data block for the specified parameter, the value FALSE is returned. Syntax errors produce an alarm. The D number cannot be set to 0 explicitly.

Example: (renaming a D number)

\[
\$TC\_DP2[1,2] = 120
\]
\[
\$TC\_DP3[1,2] = 5.5
\]
\[
\$TC\_DPCE[1,2] = 3; \text{ cutting edge number CE}
\]

... N10 def int DNoOld, DNoNew = 17 N20 DNoOld = GETDNO(1,3) N30 SETDNO(1,3,DNoNew)
This assigns cutting edge CE=3 the new D value 17. Now, these data for the cutting edge are addressed via D-number 17; both via the system parameters and in the programming with the NC address.

Additional notes

You must assign unique D numbers. Two different cutting edges of a tool must not have the same D number.

8.7.3 T numbers for the specified D number (GETACTTD)

Programming:

```
status = GETACTTD(Tno, Dno)
```

Explanation of the parameters

<table>
<thead>
<tr>
<th>Dno</th>
<th>D number to be looked for the T number.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tno</td>
<td>T number found</td>
</tr>
<tr>
<td>status</td>
<td>0: The T number was found. Tno contains the value of the T number.</td>
</tr>
<tr>
<td></td>
<td>-1: The specified D number does not have a T number; Tno=0.</td>
</tr>
<tr>
<td></td>
<td>-2: The D number is not absolute. Tno contains the value of the first tool found that contains the D number with the value Dno.</td>
</tr>
<tr>
<td></td>
<td>-5: Unable to perform the function for another reason.</td>
</tr>
</tbody>
</table>

Function

For an absolute D number, GETACTTD determines the associated T number. There is not check for uniqueness. If there are several identical D numbers within a TO unit, the T number of the first tool found is returned. If 'flat' D numbers are used, it does not make sense to use the command because the value 1 is always returned (no T number in database).
8.7.4 Set final D numbers to invalid

Programming:

```
DZERO
```

Explanation

DZERO
Marks all D number of the TO unit as invalid

Function

The command is used for support during upgrading. Offset block marked in this way are no longer checked by the language command CHKDNO.
To regain access, you must set the D number to SETDNO again
### 8.8 Toolholder kinematics

The toolholder kinematics with up to two rotary axes is programmed by means of 17 system variables $\text{TC\_CARR1}[\text{m}]$ to $\text{TC\_CARR17}[\text{m}]$. The description of the toolholder consists of:

- The vectorial distance between the first rotary axis and the toolholder reference point $l_1$, the vectorial distance between the first and the second rotary axis $l_2$, the vectorial distance between the second rotary axis and the tool reference point $l_3$.
- The reference vectors of both rotary axes $v_1$, $v_2$.
- The rotation angles $\alpha_1$, $\alpha_2$ around both axes. The rotation angles are counted in viewing direction of the rotary axis vectors, positive, in clockwise direction of rotation.

#### Resolved kinematics as of SW 5.3

For machines with resolved kinematics (both the tool and the part can rotate), the system variables have been extended to include the entries $\text{TC\_CARR18}[\text{m}]$ to $\text{TC\_CARR23}[\text{m}]$ are described as follows:

The rotatable tool table consisting of:

- The vector distance between the second rotary axis $v_2$ and the reference point of a rotatable tool table $l_4$ of the third rotary axis.

The rotary axes consisting of:

- The two channel identifiers for the reference to the rotary axes $v_1$ and $v_2$. These positions are accessed as required to determine the orientation of the orientable toolholder.

The type of kinematics with one of the values T, P or M:

- Type of kinematics T: Only tool can rotate.
- Type of kinematics P: Only part can rotate.
- Type of kinematics M: Tool and part can rotate.
### Function of the system parameter for orientable toolholders

<table>
<thead>
<tr>
<th>Designation</th>
<th>x components</th>
<th>y components</th>
<th>z components</th>
</tr>
</thead>
<tbody>
<tr>
<td>l₁ offset vector</td>
<td>$TC_CARR1[m]$</td>
<td>$TC_CARR2[m]$</td>
<td>$TC_CARR3[m]$</td>
</tr>
<tr>
<td>l₂ offset vector</td>
<td>$TC_CARR4[m]$</td>
<td>$TC_CARR5[m]$</td>
<td>$TC_CARR6[m]$</td>
</tr>
<tr>
<td>v₁ rotary axis</td>
<td>$TC_CARR7[m]$</td>
<td>$TC_CARR8[m]$</td>
<td>$TC_CARR9[m]$</td>
</tr>
<tr>
<td>v₂ rotary axis</td>
<td>$TC_CARR10[m]$</td>
<td>$TC_CARR11[m]$</td>
<td>$TC_CARR12[m]$</td>
</tr>
<tr>
<td>α₁ rotation angle</td>
<td>$TC_CARR13[m]$</td>
<td>$TC_CARR14[m]$</td>
<td></td>
</tr>
<tr>
<td>l₃ offset vector</td>
<td>$TC_CARR15[m]$</td>
<td>$TC_CARR16[m]$</td>
<td>$TC_CARR17[m]$</td>
</tr>
<tr>
<td>l₄ offset vector</td>
<td>$TC_CARR18[m]$</td>
<td>$TC_CARR19[m]$</td>
<td>$TC_CARR20[m]$</td>
</tr>
<tr>
<td>Axis identifier for rotary axis v₁</td>
<td>$TC_CARR21[m]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for rotary axis v₂</td>
<td>$TC_CARR22[m]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of kinematics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of kinematics T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of kinematics P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of Kinematics M</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset for rotary axis v₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotary axis v₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angle in degrees of rotary axes v₁ and v₂ when assuming the initial setting</td>
<td>$TC_CARR24[m]$</td>
<td>$TC_CARR25[m]$</td>
<td></td>
</tr>
<tr>
<td>Angle offset for rotary axis v₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotary axis v₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset of Hirth tooth system in degrees for rotary axes v₁ and v₂</td>
<td>$TC_CARR26[m]$</td>
<td>$TC_CARR27[m]$</td>
<td></td>
</tr>
<tr>
<td>Angle increment v₁ rotary axis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v₂ rotary axis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increment of Hirth tooth system in degrees for rotary axes v₁ and v₂</td>
<td>$TC_CARR28[m]$</td>
<td>$TC_CARR29[m]$</td>
<td></td>
</tr>
<tr>
<td>Minimum position</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotary axis v₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software limit for minimum position for rotary axes v₁ and v₂</td>
<td>$TC_CARR30[m]$</td>
<td>$TC_CARR31[m]$</td>
<td></td>
</tr>
<tr>
<td>Maximum position</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotary axis v₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software limits for maximum position for rotary axes v₁ and v₂</td>
<td>$TC_CARR32[m]$</td>
<td>$TC_CARR33[m]$</td>
<td></td>
</tr>
</tbody>
</table>
Parameters of the rotary axes from SW 6.1

The system variables are extended by the entries $TC\_CARR24[m]$ to $TC\_CARR33[m]$ and described as follows:

The **offset** of the rotary axes
- Changing the position of rotary axis $v_1$ or $v_2$ during initial setting of the orientable toolholder.

The **angle offset/angle increment** of the rotary axes
- Offset or angle increment of Hirth tooth system of rotary axes $v_1$ and $v_2$. Programmed or calculated angle is rounded up to the next value that results from $\phi = s + n \times d$ when $n$ is an integer.

The **minimum position/maximum position** of the rotary axis
- Limit angle (software limit) for rotary axis $v_1$ and $v_2$.

**Additional notes**

The number of the respective toolholder to be programmed is specified with "m.

The start/endpoints of the distance vectors on the axes can be freely selected. The rotation angles $\alpha_1$, $\alpha_2$ around the two axes are defined in the initial state of the toolholder by 0°. In this way, the kinematics of a toolholder can be programmed for any number of possibilities.

Toolholders with only one or no rotary axis at all can be described by setting the direction vectors of one or both rotary axes to zero. With a toolholder without rotary axis the distance vectors act as additional tool offsets whose components cannot be affected by a change of machining plane (G17 to G19).
Clearing the toolholder data
The data of all toolholder data sets is cleared via
$TC\_CARR1[0] = 0$.

**SW 5.3 and higher**
The type of kinematics $TC\_CARR23[T] = T$ must
be assigned one of the three permissible uppercase
or lowercase letter (T,P,M) and should not be
deleted.

Changing the toolholder data
Each of the described values can be modified by
assigning a new value in the parts program.
Any character other than T, P or M causes an alarm
when you attempt to activate the orientable
toolholder.

Reading the toolholder data
Each of the described values can be read by
assigning it to a variable in the parts program.

Supplementary conditions
A toolholder can only orientate a tool in every possible
direction in space if
- two rotary axes $v_1$ and $v_2$ are available.
- the rotary axes are positioned perpendicular to one
  another.
- the tool length axis is perpendicular to the
  second rotary axis $v_2$.

**SW 5.3 and higher**
In addition, the following requirement is applicable to
machines for which all possible orientations have to
be settable:
- Tool orientation must be perpendicular to the first
  rotary axis $v_1$. 
Programming example

The toolholder used in the following example can be fully described by a rotation around the Y axis.

N10 $TC_CARR8[1]=1  
Definition of the Y components of the first rotary axis of toolholder 1

N20 $TC_DP1[1,1]=120  
Definition of an end mill

N30 $TC_DP3[1,1]=20  
with length 20mm

N40 $TC_DP6[1,1]=5  
and with radius 5mm

N50 ROT Y37  
Frame definition with 37° rotation around the Y axis

N60 X0 Y0 Z0 F10000  
Approach initial position

N70 G42 CUT2DF TC0FR TCARR=1 T1 D1 X10  
Set radius compensation, tool length compensation in rotated frame, select toolholder 1, tool 1

N80 X40  
Execute machining under a 37° rotation

N90 Y40

N100 X0

N110 Y0

N120 M30
Path Traversing Behavior

9.1 Tangential control TANG, TANGON, TANGOF, TANGDEL ........................................ 9-352
9.2 Coupled motion TRAILON, TRAILOF ........................................................................ 9-358
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV, CTABSSV, CTABSEV ...................................................................................................................... 9-362
9.4 Axial leading value coupling, LEADON, LEADOF .................................................... 9-375
9.5 Feed characteristic, FNORM, FLIN, FCUB, FPO ....................................................... 9-381
9.6 Program run with preprocessing memory, STARTFIFO, STOPFIFO, STOPRE ......... 9-386
9.7 Repositioning on contour, REPOSA, REPOS, REPOSQ, REPOS ................. 9-388
9.1 Tangential control TANG, TANGON, TANGOF, TANGDEL

Programming

TANG (FAxis,F,LAxis1,LAxis2,Coupling,CS)
TANGON (FAxis,Angle)
TANGOF (FAxis)
TLIFT (FAxis)
TANGDEL (FAxis)

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TANG</td>
<td>Preparatory instruction for the definition of a tangential follow-up</td>
</tr>
<tr>
<td>TANGON</td>
<td>Activate tangential control specifying following axis and offset angle</td>
</tr>
<tr>
<td>TANGOF</td>
<td>Deactivate tangential control specifying following axis</td>
</tr>
<tr>
<td>TLIFT</td>
<td>Insert intermediate block at contour corners</td>
</tr>
<tr>
<td>TANGDEL</td>
<td>Delete definition of a tangential follow-up</td>
</tr>
</tbody>
</table>

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAxis</td>
<td>Following axis: additional tangential following rotary axis</td>
</tr>
<tr>
<td>LAxis1, LAxis2</td>
<td>Leading axes: path axes which determine the tangent for the following axis</td>
</tr>
<tr>
<td>Coupling</td>
<td>Coupling factor: relationship between the angle change of the tangent and the following axis. Parameter optional; default: 1</td>
</tr>
<tr>
<td>CS</td>
<td>Identifier for coordinate system</td>
</tr>
<tr>
<td>Angle</td>
<td>Offset angle of following axis</td>
</tr>
</tbody>
</table>

"B" = basic coordinate system; data optional; default
"W" = workpiece coordinate system
9.1 Tangential control TANG, TANGON, TANGOF, TANGDEL

Function

A rotary axis (= following axis) follows the programmed path of two leading axes. The following axis is located at a defined offset angle to the path tangent.

Applications

Tangential control can be used in applications such as:

- Tangential positioning of a rotatable tool during nibbling
- Follow-up of the tool orientation on a band saw
- Positioning of a dresser tool on a grinding wheel (see diagram)
- Positioning of a cutting wheel for glass or paper working
- Tangential infeed of a wire in five-axis welding

Sequence

Defining following axis and leading axis

TANG is used to define the following and leading axes.

A coupling factor specifies the relationship between an angle change on the tangent and the following axis. Its value is generally 1 (default).

The follow-up can be performed in the basic coordinate system "B" (default) or the workpiece coordinate system "W".

Example:

```
TANG(C, X, Y, 1, "B")
```

Meaning:

Rotary axis C follows geometry axes X and Y.
Activating/deactivating tangential control:

**TANGON, TANGOF**

Tangential control is called with TANGON specifying the following axis and the desired offset angle of the following axis:

\[ \text{TANGON}(C, 90) \]

**Meaning:**

C axis is the following axis. On every movement of the path axes, it is rotated into a position at 90° to the path tangent.

The following axis is specified in order to deactivate the tangential control:

\[ \text{TANGOF}(C) \]

**Angle limit through working area limitation**

For path movements which oscillate back and forth, the tangent jumps through 180° at the turning point on the path and the orientation of the following axis changes accordingly. This behavior is generally inappropriate: the return movement should be traversed at the same negative offset angle as the approach movement.

This can be achieved by limiting the working area of the following axis (G25, G26). The working area limitation must be active at the instant of path reversal (WALIMON).

If the offset angle lies outside the working area limit, an attempt is made to return to the permissible working area with the negative offset angle.
Insert intermediate block at contour corners, TILIFT

At one corner of the contour the tangent changes and thus the setpoint position of the following axis. The axis normally tries to compensate this step change at its maximum possible velocity. However, this causes a deviation from the desired tangential position over a certain distance on the contour after the corner. If such a deviation is unacceptable for technological reasons, the instruction TILIFT can be used to force the control to stop at the corner and to turn the following axis to the new tangent direction in an automatically generated intermediate block. The axis is rotated at its maximum possible velocity.

The TILIFT(...) instruction must be programmed immediately after the axis assignment with TANG(...).

Example:
TANG(C,X,Y...)
TILIFT(C)

Deactivate TILIFT

To deactivate TILIFT, repeat the axis assignment TANG(...) without inserting TILIFT(...) afterwards.

The angular change limit at which an intermediate block is automatically inserted is defined via machine data $MA_EPS_TLIFT_TANG_STEP.

Delete definition of a tangential follow-up

An existing user-defined tangential follow-up must be deleted if a new tangential follow-up with the same following axis is defined in the preparation call TANG.
### 9.1 Tangential control TANG, TANGON, TANGOF, TANGDEL

**TANGDEL (FAxis)**
Delete tangential follow-up

Deletion is only possible if the coupling with TANGOF(Faxis) is deactivated.

**Programming example**

**Example of plane change**

1st definition of the tang. follow-up
Activation of the coupling

1st definition of the tang. follow-up
Activation of the new coupling

An alarm is output.

Deactivation of follow-up with Y1
Delete 1st definition

Deactivation of follow-up with Y1
Delete 1st definition

Programming example

With geometry axis switchover and TANGDEL

Y1 is geo axis 2

Y2 is the new geo axis 2

2nd definition of the tang. follow-up
Activation of the follow-up with 2nd def.
### Additional notes

**Influence on transformations**
The position of the following rotary axis can be an input value for a transformation.

**Explicit positioning of the following axis**
If an axis which is following your lead axes is positioned explicitly the position is added to the programmed offset angle.
All path definitions are possible: Path and positioning axis movements.

**Coupling status**
You can query the status of the coupling in the NC program with the following system variable:

\[ $AA\_COUP\_ACT[\text{Axis}] \]

- **0** No coupling active
- **1,2,3** Tangential follow-up active
9.2 Coupled motion TRAILON, TRAILOF

Programming

TRAILON (FAxis, LAxis, Coupling)
TRAILOF (FAxis, LAxis, Axis2)

Explanation of the commands and parameters

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAILON</td>
<td>Activate and define coupled axes; modal</td>
</tr>
<tr>
<td>TRAILOF</td>
<td>Deactivate coupled axes</td>
</tr>
<tr>
<td>FAxis</td>
<td>Axis name of trailing axis</td>
</tr>
<tr>
<td>LAxis</td>
<td>Axis name of leading axis</td>
</tr>
<tr>
<td>Coupling</td>
<td>Coupling factor = Path of coupled-motion axis/path of trailing axis</td>
</tr>
<tr>
<td></td>
<td>Default = 1</td>
</tr>
</tbody>
</table>

Function

When a defined leading axis is moved, the trailing axes (= following axes) assigned to it traverse through the distances described by the leading axis, allowing for a coupling factor.
Together, the leading axis and following axis represent coupled axes.

Applications

- Traversing of an axis by a simulated axis. The leading axis is a simulated axis and the trailing axis is a real axis. The real axis can thus be traversed with allowance for the coupling factor.
- Two-sided machining with 2 combined axis pairs:
  1st leading axis Y, trailing axis V
  2nd leading axis Z, trailing axis W
9.2 Coupled motion TRAILON, TRAILOF

Sequence

Defining coupled-axis combinations, TRAILON

The coupled axes are defined and activated simultaneously with the modal language command TRAILON.

TRAILON(V,Y)

V = trailing axis, Y = leading axis

The number of coupled axes that can be activated simultaneously is restricted only by the possible combinations of axes on the machine.

Coupled motion always takes place in the basic coordinate system (BCS).

Coupled axis types

A coupled-axis group can consist of any combination of linear and rotary axes. A simulated axis can also be defined as a leading axis.

Coupled-motion axes

Up to two leading axes can be assigned simultaneously to a trailing axis. The assignment is made in different combinations of coupled axes.

A trailing axis can be programmed with all the available motion commands (G0, G1, G2, G3, ...). In addition to paths defined independently, the trailing axis also traverses the distances derived from its leading axes, allowing for the coupling factors.

A trailing axis can also act as a leading axis for other trailing axes. Various combinations of coupled axes can be set up in this way.
9.2 Coupled motion TRAILON, TRAILOF

**Coupling factor**
The coupling factor specifies the desired ratio of the paths of trailing axis and leading axis.

\[
\text{Coupling factor} = \frac{\text{Path of trailing axis}}{\text{Path of leading axis}}
\]

If the coupling factor is not specified in the program, a coupling factor of 1 is automatically taken as the default.

The factor is entered as a decimal fraction (type REAL). The input of a negative value causes opposite traversing movements on the leading and trailing axes.

**Deactivate coupled axes**
The following language command deactivates the coupling with a leading axis:

\[
\text{TRAILOF}(V,Y)
\]

\(V = \text{trailing axis}, Y = \text{leading axis}\)

TRAILOF with 2 parameters deactivates the coupling to only 1 leading axis.

If a trailing axis is assigned to 2 leading axes, e.g. \(V=\text{trailing axis} \text{ and } X,Y=\text{leading axes}\), TRAILOF can be called with 3 parameters to deactivate the coupling:

\[
\text{TRAILOF}(V,X,Y)
\]
**9.2 Coupled motion TRAILON, TRAILOF**

Additional notes

**Acceleration and velocity**
The acceleration and velocity limits of the combined axes are determined by the "weakest axis" in the combined axis pair.

**Coupling status**
You can query the status of the coupling in the NC program with the following system variable:

\[ $AA\_COUP\_ACT[axis]$ \]

- 0  No coupling active
- 8  Coupled motion active

**Programming example**
The workpiece is to be machined on two sides with the axis configuration shown in the diagram. To do this, you create 2 combinations of coupled axes.

```
N100 TRAILON(V,Y)  Activate 1st combined axis pair
N110 TRAILON(W,Z,-1) Activate 2nd combined axis pair, coupling factor negative: trailing axis traverses in opposite direction to leading axis
N120 G0 Z10        Infeed of Z and W axes in opposite axis directions
N130 G0 Y20        Infeed of Y and V axes in same axis directions
...
N200 G1 Y22 V25 F200 Superimpose dependent and independent movement of trailing axis "V"
...
TRAILOF(V,Y)       Deactivate 1st coupled axis
TRAILOF(W,Z)       Deactivate 2nd coupled axis
```
Programming

The following modal NC commands work with curve tables:
(You will find explanations of the parameters at the end of the list of functions.)

A) Main functions
Curve tables are defined in a parts program.

CTABDEF (Faxis,Laxis,n,applim, memType)  Define beginning of curve table
CTABEND ()  Define end of curve table
CTABDEL (n)  Delete a curve table
CTABDEL ()  Deletion of all curve tables, independently of memType
CTABDEL (n, m)  Deletion of a curve table range
CTABDEL (n, m, memType)  Deletion of the curve tables of the curve table range that are stored in memType
CTABDEL (, , memType)  Deletion of all curve tables in the specified memory

R10=CTAB(LW,n,degrees,FAxis,LAxis)  Following value for a leading value
R10=CTABINV(FW,aproxLW,n,degrees,FAxis,LAxis)  Leading value to a following value
R10=CTABSSV(LV,n,degree,Faxis,Laxis)  Starting value of the following axis in the segment belonging to the LV
R10=CTABSEV(LV,n,degree,Faxis,Laxis)  End value of the following axis in the segment belonging to the LV

General form:
CTABLOCK (n, m, memType)  Set a lock against deletion or overwriting.

CTABLOCK (n)  Applications in the forms:
CTABLOCK (n, m)  Curve table with number n
CTABLOCK ()  Curve tables in number range n to m
CTABLOCK (, , memType)  All curve tables irrespective of memory type
CTABLOCK (, , memType)  All curve tables in the specified memory type
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV

General form:

\[ \text{CTABUNLOCK}(n, m, \text{memType}) \]

Cancel a lock against deletion or overwriting.

CTABUNLOCK enables the tables disabled with CTABLOCK. Tables that function in an active coupling remain disabled, i.e. they still cannot be deleted. But the CTABLOCK lock is canceled, i.e., as soon as locking via the active coupling is canceled by deactivating the coupling, this table can be deleted. It is not necessary to call CTABUNLOCK again.

Applications in the forms:

- Curve table with number \( n \)
- Curve tables in number range \( n \) to \( m \)
- All curve tables irrespective of memory type
- All curve tables in the specified memory type
For further information about leading and following values, see Section "Axial leading value coupling" and "Path leading value coupling" in this section.

Additional functions exist for diagnostics and optimization of resource use. These are described in the M3 Description of Functions.

### Explanation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAxis</td>
<td>Following axis: Axis that is programmed via the curve table.</td>
</tr>
<tr>
<td>LAxis</td>
<td>Leading axis. Axis that is programmed with the leading value.</td>
</tr>
<tr>
<td>n, m</td>
<td>Number of the curve table; n &lt; m in CTABDEL(n, m) The number of the curve table is unique and not dependent on the memory type. Tables with the same number can be in the SRAM and DRAM.</td>
</tr>
<tr>
<td>p</td>
<td>Entry location (in memory range memType)</td>
</tr>
</tbody>
</table>
| applim | Identifier for table periodicity:  
0 Table is not periodic  
1 Table is periodic with regard to the leading axis  
2 Table is periodic with regard to leading axis and following axis |
| LW     | Leading value. Positional value of the leading axis for which a following value is to be calculated. |
| degrees| Parameter name for gradient parameter |
| FW     | Following value. Positional value of the following axis for which a leading value is to be calculated. |
| aproxLW| Approximation solution for leading value if no specific leading value can be determined for a following value. |
| FAxis,LAxis | Optional specification of the following and/or leading axis |
| memType | Optional specification of memory type of the NC: "DRAM" / "SRAM" If no value is programmed for this parameter, the default memory type set in MD 20905: CTAB_DEFAULT_MEMORY_TYPE is used. |
11.02 Path Traversing Behavior

9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV

Function

You can use curve tables to program position and velocity relationships between 2 axes.

**Example** of substitution of mechanical cam: The curve table forms the basis for the axial leading value coupling by creating the functional relationship between the leading and the following value: With appropriate programming, the control calculates a polynomial that corresponds to the cam plate from the relative positions of the leading and following axes.

Additional notes

To create curve tables the memory space must be reserved by setting the machine data.

Definition of a curve table

CTABDEF, CTABEND

A curve table represents a parts program or a section of a parts program which is enclosed by CTABDEF at the beginning and CTABEND at the end.

Within this parts program section, unique following axis positions are assigned to individual positions of the leading axis by traverse statements and used as intermediate positions in calculating the curve definition in the form of a polynomial up to the 3rd order.

As from SW 6, intermediate points for curve definitions can be calculated in the form of an up to 5th order polynomial.
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV,

Starting and end value of the curve table:
The starting value for the beginning of the definition range of the curve table are the first associated axis positions specified (the first traverse statement) within the curve table definition. The end value of the definition range of the curve table is determined in accordance with the last traverse command.

Within the definition of the curve table, you have use of the entire NC language.

Additional notes
The following are not permissible:
- Preprocess stop
- Jumps in the leading axis movement (e.g. on changing transformations)
- Traverse statement for the following axis only
- Reversal of the leading axis, i.e. position of the leading axis must always be unique
- CTABDEF and CTABEND statement on various program levels.

SW 6.3
Depending on MD 20900 CTAB_ENABLE_NO_LEADMOTION, jumps of the following axis can be tolerated if leading axis motion is missing. The other restrictions give in the notice still apply.
Specification of the NC memory type can be used in table creation and deletion.

All modal statements that are made within the curve table definition are invalid when the table definition is completed. The parts program in which the table definition is made is therefore located in front of and after the table definition in the same state.

R parameter assignments are reset.
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV

Example:

...  
R10=5 R11=20  
...
CTABDEF  
G1 X=10 Y=20 F1000  
R10=R11+5 ;R10=25  
X=R10  
CTABEND  
... ;R10=5

Repeated use of curve tables

The function relation between the leading axis and the following axis calculated through the curve table is retained under the table number beyond the end of the parts program and during power-off. The curve table created can be applied to any axis combinations of leading and following axes whatever axes were used to create the curve table.
Behavior at the edges of the curve table

**Non-periodic** curve table
If the leading value is outside the definition range, the following value output is the upper or lower limit.

**Periodic** curve table
If the leading value is outside the definition range, the leading value is evaluated modulo of the definition range and the corresponding following value is output.
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV.

Reading table positions, CTAB, CTABINV

With CTAB you can read the following value for a leading value directly from the parts program or from synchronized actions (Chapter 10).

With CTABINV, you can read the leading value for a following value. This assignment does not always have to be unique. CTABINV therefore requires an approximate value (aproxLW) for the expected leading value. CTABINV returns the leading value that is closest to the approximate value. The approximate value can be the leading value from the previous interpolation cycle.

Both functions also output the gradient of the table function at the correct position to the gradient parameter (degrees). In this way, the you can calculate the speed of the leading or following axis at the corresponding position.

Reading segment positions, CTABSSV, CTABSEV

CTABSSV can be used to read the starting value of the curve segment belonging to the specified leading value directly from the parts program or from synchronous actions (Chapter 10).

CTABSEV can be used to read the end value of the curve segment belonging to the specified leading value directly from the parts program or from synchronous actions (Chapter 10).

Additional notes

Optional specification of the leading or following axis for CTAB/CTABINV/CTABSSV/CTABSEV is important if the leading and following axes are configured in different length units.
Programming example

Use of CTABSSV and CTABSEV

N10 DEF REAL STARTPOS
N20 DEF REAL ENDPOS
N30 DEF REAL GRADIENT
...
N100 CTABDEF(Y,X,1,0)
N110 X0 Y0
N120 X20 Y10
N130 X40 Y40
N140 X60 Y10
N150 X80 Y0
N160 CTABEND
...
N200 STARTPOS = CTABSSV(30.0, 1, GRADIENT)
...
N210 ENDPOS = CTABSEV(30.0, 1, GRADIENT)

Beginning of table definition
Starting position 1st table segment
End position 1st table segment = start position 2nd table segment ...

End of table definition
Start position Y in segment 2 = 10
End position Y in segment 2 = 40
Segment 2 belongs to LV X = 30.0.

Deleting curve tables, CTABDEL

With CTABDEL you can delete the curve tables. Curve tables that are active in a coupling cannot be deleted. If at least one curve table is active out of a multiple delete command CTABDEL() or CTABDEL(n, m) in a coupling, none of the addressed curve tables will be deleted. As from SW 6.3, curve tables of a certain memory type can be deleted by optional memory type specification.
9.3 Curve tables, CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV

Overwriting curve tables
A curve table is overwritten as soon as its number is used in another table definition. Active tables cannot be overwritten.

Additional notes
No warning is output when you overwrite curve tables!

Additional notes
With the system variable $P_CTABDEF it is possible to query from inside a parts program whether a curve table definition is active.

The parts program section can be used as a curve table definition after excluding the statements and therefore as a real parts program again.

Programming example
A program section is to be used unchanged for defining a curve table. The command for preprocess stop STOPRE can remain and is active again immediately as soon as the program section is not used for table definition and CTABDEF and CTABEND have been removed:

```
CTABDEF(Y,X,1,1)
...
...
IF NOT ($P_CTABDEF)
STOPRE
ENDIF
...
...
CTABEND
```
Curve tables and various operating states

During active block search, calculation of curve tables is not possible. If the target block is within the definition of a curve table, an alarm is output when CTABEND is reached.

Programming example 1

Definition of a curve table

N100 CTABDEF(Y,X,3,0)  
Beginning of the definition of a non-periodic curve table with number 3

N110 X0 Y0
1. Traverse statement defines starting values and 1st intermediate point: Leading value: 5; Following value: 0

N120 X20 Y0
2. Intermediate point: Leading value: 0...20; Following value: Starting value...0

N130 X100 Y6
3. Intermediate point: Leading value: 20...100; Following value: 0...6

N140 X150 Y6
4. Intermediate point: Leading value: 100...150; Following value: 6...6

N150 X180 Y0
5. Intermediate point: Leading value: 150...180; Following value: 6...0

N200 CTABEND
End of the definition; The curve table is generated in its internal representation as a polynomial up to the 3rd order;
The calculation of the curve definition depends on the modally selected interpolation type (circle, linear, spline interpolation); The parts program state before the beginning of the definition is restored.

Programming example 2

Definition of a periodic curve table with number 2, leading value range 0 to 360, following axis motion from 0 to 45 and back to 0:

N10 DEF REAL DEPPOS;
N20 DEF REAL GRADIENT;
N30 CTABDEF(Y,X,2,1)  
Beginning of definition
N40 G1 X=0 Y=0
N50 POLY
N60 PO[X]=(45.0)
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)
N80 PO[X]=(270.0)
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)
N100 PO[X]=(360.0)
N110 CTABEND  
End of definition

Test of the curve by coupling Y to X:
N120 G1 F1000 X0
N130 LEADON(Y,X,2)
N140 X360
N150 X0
N160 LEADOF(Y,X)

Read the table function for leading value 75.0:
N170 DEPPOS=CTAB(75.0,2,GRADIENT)

Positioning of the leading and the following axis:
N180 G0 X75 Y=DEPPOS
After activating the coupling no synchronization of the following axis is required:

N190 LEADON(Y,X,2)
N200 G1 X110 F1000
N210 LEADOF(Y,X)
N220 M30
9.4 Axial leading value coupling, LEADON, LEADOF

**Programming**

LEADON(FAxis,LAxis,n)
LEADOF(FAxis,LAxis,n)

**Explanation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEADON</td>
<td>Activate leading value coupling</td>
</tr>
<tr>
<td>LEADOF</td>
<td>Deactivate leading value coupling</td>
</tr>
<tr>
<td>FAxis</td>
<td>Following axis</td>
</tr>
<tr>
<td>LAxis</td>
<td>Leading axis</td>
</tr>
<tr>
<td>n</td>
<td>Curve table number</td>
</tr>
</tbody>
</table>

**Function**

With the axial leading value coupling, a leading and a following axis are moved in synchronism. It is possible to assign the position of the following axis via a curve table or the resulting polynomial uniquely to a position of the leading axis – simulated if necessary.

**Leading axis** is the axis which supplies the input values for the curve table. **Following axis** is the axis which takes the positions calculated by means of the curve table.

The leading value coupling can be activated and deactivated both from the parts program and during the movement from synchronized actions (Chapter 10).

The leading value coupling always applies in the basic coordinate system.

For information about creating curve table, see Chapter "Curve tables" in this chapter. For information about leading value coupling, see /FB/, M3, Coupled Motion and Leading Value Coupling.
## Sequence

Leading value coupling requires synchronization of the leading and the following axes. This synchronization can only be achieved if the following axis is inside the tolerance range of the curve definition calculated from the curve table when the leading value coupling is activated.

The tolerance range for the position of the following axis is defined via machine data 37200 COUPLE_POS_TOL_COARSE.

If the following axis is not yet at the correct position when the leading value coupling is activated, the synchronization run is automatically initiated as soon as the position setpoint value calculated for the following axis is approximately the real following axis position. During the synchronization procedure the following axis is traversed in the direction that is defined by the setpoint speed of the following axis (calculated from master spindle and CTAB).

### Additional notes

If the following axis position calculated moves away from the current following axis position when the leading value coupling is activated, it is not possible to establish synchronization.

### Actual value and setpoint coupling

The following can be used as the leading value, i.e. as the output values for position calculation of the following axis:

- Actual values of the leading axis position: Actual value coupling
- Setpoints of the leading axis position: Setpoint coupling
Additional notes

Setpoint coupling provides better synchronization of the leading and following axis than actual value coupling and is therefore set by default.

Setpoint coupling is only possible if the leading and following axis are interpolated by the same NCU. With an external leading axis, the following axis can only be coupled to the leading axis via the actual values.

**Switchover** between actual and setpoint coupling

A switchover can be programmed via setting data $SA_LEAD_TYPE

You must always switch between the actual-value and setpoint coupling when the following axis stops. It is only possible to resynchronize after switchover when the axis is motionless.

**Application example:**

You cannot read the actual values without error during large machine vibrations. If you use leading value coupling in press transfer, it might be necessary to switchover from actual-value coupling to setpoint coupling in the work steps with the greatest vibrations.
9.4 Axial leading value coupling, LEADON, LEADOF

Leading value simulation with setpoint simulation

Via machine data, you can disconnect the interpolator for the leading axis from the servo. In this way you can generate setpoints for setpoint coupling without actually moving the leading axis.

Leading values generated from a setpoint link can be read from the following variables so that they can be used, for example, in synchronized actions:
- `$AA_LEAD_P` Leading value position
- `$AA_LEAD_V` Leading value velocity

Additional notes

As an option, leading values can be generated with other self-programmed methods. The leading values generated in this way are written into the variables
- `$AA_LEAD_SP` Leading value position
- `$AA_LEAD_SV` Leading value velocity

and read from them. Before you use these variables, setting data `$SA_LEAD_TYPE = 2` must be set.

Status of coupling

You can query the status of the coupling in the NC program with the following system variable:

```
$AA_COUP_ACT[axis]
```

- 0 No coupling active
- 16 Leading value coupling active

Deactivate leading value coupling, LEADOF

When you deactivate the leading value coupling, the following axis becomes a normal command axis again!

Axial leading value coupling and different operating states

Depending on the setting in the machine data, the leading value couplings are deactivated with RESET.
Programming example

In a pressing plant, an ordinary mechanical coupling between a leading axis (stanchion shaft) and axis of a transfer system comprising transfer axes and auxiliary axes is to be replaced by an electronic coupling system.

It demonstrates how a mechanical transfer system is replaced by an electronic transfer system. The coupling and decoupling events are implemented as static synchronized actions.

From the leading axis LW (stanchion shaft), transfer axes and auxiliary axes are controlled as following axes that are defined via curve tables.

**Following axes**

<table>
<thead>
<tr>
<th>Axis</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Feed or longitudinal axis</td>
</tr>
<tr>
<td>YL</td>
<td>Closing or lateral axis</td>
</tr>
<tr>
<td>ZL</td>
<td>Stroke axis</td>
</tr>
<tr>
<td>U</td>
<td>Roller feed, auxiliary axis</td>
</tr>
<tr>
<td>V</td>
<td>Guiding head, auxiliary axis</td>
</tr>
<tr>
<td>W</td>
<td>Greasing, auxiliary axis</td>
</tr>
</tbody>
</table>

**Status management**

Switching and coupling events are managed via real-time variables:

$$AC\_MARKER[i]=n$$  
with:  
i     Marker number  
n     Status value

**Actions**

The actions that occur include, for example, the following synchronized actions:

- Activate coupling, LEADON(following axis, leading axis, curve table number)
- Deactivate coupling, LEADOFFollowing axis, leading axis)
- Set actual value, PRESETON(axis, value)
- Set marker, $$AC\_MARKER[i]=value$$
- Coupling type: real/virtual leading value
- Approaching axis positions, POS[axis]=value

**Conditions**

Fast digital inputs, real-time variables $$AC\_MARKER$$ and position comparisons are linked using the Boolean operator AND for evaluation as conditions.

**Note**

In the following example, line change, indentation and **bold** type are used for the sole purpose of improving readability of the program. To the controller, everything that follows a line number constitutes a single line.
Comment

; Defines all static synchronized actions.
; **** Reset marker

N2 $AC_MARKER[0]=0 $AC_MARKER[1]=0

; **** E1 0=>1 Coupling transfer ON
N10 IDS=1 EVERY ($A_IN[1]==1) AND ($A_IN[16]==1) AND ($AC_MARKER[0]==0)
DO LEADON(X,LW,1) LEADON(YL,LW,2)
    LEADON(ZL,LW,3) $AC_MARKER[0]=1

; **** E1 0=>1 Coupling roller feed ON
DO LEADON(U,LW,4) PRESETON(U,0)
    $AC_MARKER[5]=1

; **** E1 0=>1 Coupling guide head ON
DO LEADON(V,LW,4) PRESETON(V,0)
    $AC_MARKER[6]=1

; **** E1 0=>1 Coupling greasing ON
DO LEADON(W,LW,4) PRESETON(W,0)
    $AC_MARKER[7]=1

; **** E2 0=>1 Coupling OFF
N30 IDS=3 EVERY ($A_IN[2]==1)
DO LEADOF(X,LW) LEADOF(YL,LW)
    LEADOF(ZL,LW) LEADOF(U,LW)
    LEADOF(V,LW) LEADOF(W,LW) $AC_MARKER[0]=0
    $AC_MARKER[1]=0 $AC_MARKER[3]=0

....
N110 G04 F01
N120 M30
9.5 Feed characteristic, FNORM, FLIN, FCUB, FPO

**Programming**

- \( F \)... FNORM
- \( F \)... FLIN
- \( F \)... FCUB
- \( F = F^{\text{PO}}(\ldots, \ldots, \ldots) \)

**Explanation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNORM</td>
<td>Basic setting. The feed value is specified as a function of the traverse path of the block and is then valid as a modal value.</td>
</tr>
<tr>
<td>FLIN</td>
<td>Path velocity profile linear: The feed value is approached linearly via the traverse path from the current value at the block beginning to the block end and is then valid as a modal value.</td>
</tr>
<tr>
<td>FCUB</td>
<td>Path velocity profile cubic: The non-modally programmed ( F ) values are connected by means of a spline referred to the block end point. The spline begins and ends tangentially with the previous and the following feedrate specification. If the ( F ) address is missing from a block, the last ( F ) value to be programmed is used.</td>
</tr>
<tr>
<td>( F = F^{\text{PO}} )</td>
<td>Polynomial path velocity profile: The ( F ) address defines the feed characteristic via a polynomial from the current value to the block end. The end value is valid thereafter as a modal value.</td>
</tr>
</tbody>
</table>

**Function**

To permit flexible definition of the feed characteristic, the feed programming according to DIN 66205 has been extended by linear and cubic characteristics. The cubic characteristics can be programmed either directly or as interpolating splines. These additional characteristics make it possible to program continuously smooth velocity characteristics depending on the curvature of the workpiece to be machined. These additional characteristics make it possible to program continuously smooth velocity characteristics depending on the curvature of the workpiece to be machined.
Sequence

**FNORM**
The feed address F defines the path feed as a constant value according to DIN 66025.

Please refer to Programming Guide "Fundamentals" for more detailed information on this subject.

**FLIN**
The feed characteristic is approached linearly from the current feed value to the programmed F value until the end of the block.

Example:
N30 F1400 FLIN X50
**9.5 Feed characteristic, FNORM, FLIN, FCUB, FPO**

**FCUB**
The feed is approached according to a cubic characteristic from the current feed value to the programmed F value until the end of the block. The control uses splines to connect all the feed values programmed non-modally that have an active FCUB. The feed values act here as interpolation points for calculation of the spline interpolation.

Example:
N50 F1400 FCUB X50
N60 F2000 X47
N70 F3800 X52
...

\[ F = F_{PO}(\ldots, \ldots, \ldots) \]

The feed characteristic is programmed directly via a polynomial. The polynomial coefficients are specified according to the same method used for polynomial interpolation.

Example:
\[ F = F_{PO}(\text{endfeed}, \text{quadf}, \text{cubf}) \]

`endfeed`, `quadf` and `cubf` are previously defined variables.

---

**endfeed**: Feed at block end

**quadf**: Quadratic polynomial coefficient

**cubf**: Cubic polynomial coefficient

With an active FCUB, the spline is linked tangentially to the characteristic defined via FPO at the block beginning and block end.

**Supplementary conditions**
The functions for programming the path traversing characteristics apply regardless of the programmed feed characteristic.
The programmed feed characteristic is always absolute regardless of G90 or G91.

### Additional notes

**Compressor**

With an active compressor COMPON the following applies when several blocks are joined to form a spline segment:

- **FNORM:**
  The F word of the last block in the group applies to the spline segment.

- **FLIN:**
  The F word of the last block in the group applies to the spline segment.
  The programmed F value applies until the end of the segment and is then approached linearly.

- **FCUB:**
  The generated feed spline deviates from the programmed end points by an amount not exceeding the value set in machine data 
  $\text{SMC\_COMPESS\_VELO\_TOL}$

  $F = \text{FPO}(\ldots,\ldots)\\$
  These blocks are not compressed.

**Feed optimization on curved path sections**

Feed polynomial F-FPO and feed spline FCUB should always be traversed at constant cutting rate CFC, thereby allowing a jerk-free setpoint feed profile to be generated. This enables creation of a continuous acceleration setpoint feed profile.
Programming example

This example shows you the programming and graphic representation of various feed profiles.

N1 F1000 FNORM G1 X8 G91 G64  Constant feed profile, incremental dimensioning
N2 F2000 X7  Step change in setpoint velocity
N3 F=FPO(4000, 6000, -4000)  Feed profile via polynomial with feed 4000 at block end
N4 X6  Polynomial feed 4000 applies as modal value
N5 F3000 FLIN X5  Linear feed profile
N6 F2000 X8  Linear feed profile
N7 X5  Linear feed applies as modal value
N8 F1000 FNORM X5  Constant feed profile with abrupt change in acceleration rate
N9 F1400 FCUB X8  All subsequent, non-modally programmed F values are connected via splines
N10 F2200 X6
N11 F3900 X7
N12 F4600 X7
N13 F4900 X5  Deactivate spline profile
N14 FNORM X5
N15 X20
9.6 Program run with preprocessing memory, STARTFIFO, STOPFIFO, STOPRE

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOPFIFO</td>
<td>Stop high-speed processing section, fill preprocessing memory, until STARTFIFO, &quot;Preprocessing memory full&quot; or &quot;End of program&quot; is detected.</td>
</tr>
<tr>
<td>STARTFIFO</td>
<td>Start of high-speed processing section, in parallel to filling the preprocessing memory</td>
</tr>
<tr>
<td>STOPRE</td>
<td>Preprocessing stop</td>
</tr>
</tbody>
</table>

Function

Depending on its expansion level, the control system has a certain quantity of so-called preprocessing memory in which prepared blocks are stored prior to program execution and then output as high-speed block sequences while machining is in progress. These sequences allow short paths to be traversed at a high velocity.

Provided that there is sufficient residual control time available, the preprocessing memory is always filled. STARTFIFO stops the machining process until the preprocessing memory is full or until STOPFIFO or STOPRE is detected.

Sequence

Mark processing section

The high-speed processing section to be buffered in the preprocessing memory is marked at the beginning and end with STARTFIFO and STOPFIFO respectively.

Example:

```
N10 STOPFIFO
N20...
N100
N110 STARTFIFO
```

Execution of these blocks does not begin until the preprocessing memory is full or command STARTFIFO is detected.
Restrictions
The preprocessing memory is not filled or the filling process interrupted if the processing section contains commands that require unbuffered operation (reference point approach, measuring functions, ...).

Stop preprocessing
When STOPRE is programmed, the following block is not processed until all previously prepared and stored blocks have been fully executed. The previous block is halted with exact stop (as for G9).

Example:
N10 ...
N30 MEAW=1 G1 F1000 X100 Y100 Z50
N40 STOPRE

The control system initiates an internal preprocessing stop while status data of the machine ($A...$) are accessed.

Example:
R10 = $AA_IM[X]$ ; Read actual value of X axis

Note
When a tool offset or spline interpolations are active, you should not program the STOPRE command as this will lead to interruption in contiguous block sequences.
9.7 Repositioning on contour, REPOSA, REPOSL, REPOSQ, REPOSH

**Programming**

- **REPOSA** RMI DISPR=... or REPOSA RMB or REPOSA RME
- **REPOSL** RMI DISPR=... or REPOSL RMB or REPOSL RME
- **REPOSQ** RMI DISPR=... DISR=... or REPOSQ RMB DISR=... or REPOSQ RME DISR=... or REPOSQA DISR=...
- **REPOSH** RMI DISPR=... DISR=... or REPOSH RMB DISR=... or REPOSH RME DISR=... or REPOSHA DISR=...

**Explanation of the commands**

### Approach path

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPOSA</td>
<td>Approach along line on all axes</td>
</tr>
<tr>
<td>REPOSL</td>
<td>Approach along line</td>
</tr>
<tr>
<td>REPOSQ DISR=...</td>
<td>Approach along quadrant with radius DISR</td>
</tr>
<tr>
<td>REPOSQA DISR=...</td>
<td>Approach on all axes along quadrant with radius DISR</td>
</tr>
<tr>
<td>REPOSH DISR=...</td>
<td>Approach along semi-circle with diameter DISR</td>
</tr>
<tr>
<td>REPOSHA DISR=...</td>
<td>Approach on all axes along semi-circle with diameter DISR</td>
</tr>
</tbody>
</table>

### Repositioning point

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMI</td>
<td>Approach interruption point</td>
</tr>
<tr>
<td>RMI DISPR=...</td>
<td>Entry point at distance DISPR in mm/inch in front of interruption point</td>
</tr>
<tr>
<td>RMB</td>
<td>Approach block start point</td>
</tr>
<tr>
<td>RME DISPR=...</td>
<td>Approach block end point at distance DISPR in front of end point</td>
</tr>
<tr>
<td>A0 B0 C0</td>
<td>Axes in which approach is to be made</td>
</tr>
</tbody>
</table>
**Function**

If you interrupt the program run and retract the tool during the machining operation because, for example, the tool has broken or you wish to check a measurement, you can reposition at any selected point on the contour under control by the program.

The REPOS command acts in the same way as a subprogram return jump (e.g. via M17). Blocks programmed after the command in the interrupt routine are not executed.

For information about interrupting program runs, see also Section "Interrupt routine" in Programming Guide "Advanced".

**Sequence**

**Defining repositioning point**

With reference to the NC block in which the program run has been interrupted, it is possible to select one of three different repositioning points:

- RMI, interruption point
- RMB, block start point or last end point
- RME, block end point

RMI DISPR=... or RME DISPR=... allows you to select a repositioning point which sits before the interruption point or the block end point.

DISPR=... allows you to describe the contour distance in mm/inch between the repositioning point and the interruption before the end point. Even for high values, this point cannot be further away than the block start point.

If no DISPR=... command is programmed, then DISPR=0 applies and with it the interruption point (with RMI) or the block end point (with RME).

**SW 5.2 and higher:**

The sign before DISPR is evaluated.

In the case of a plus sign, the behavior is as previously.
In the case of a minus sign, approach is behind the interruption point or, with RMB, behind the block start point. The distance between interruption point and approach point depends on the value of DISPR. Even for higher values, this point can lie in the block end point at the maximum. **Application example:** A sensor will recognize the approach to a clamp. An ASUB is initiated to bypass the clamp. Afterwards, a negative DISPR is repositioned on one point behind the clamp and the program is continued.

**Approach with new tool**
The following applies if you have stopped the program run due to tool breakage: When the new D number is programmed, the machining program is continued with modified tool offset values at the repositioning point.

Where tool offset values have been modified, it may not be possible to reapproach the interruption point. In such cases, the point closest to the interruption point on the new contour is approached (possibly modified by DISPR).

**Approach contour**
The motion with which the tool is repositioned on the contour can be programmed. Enter zero for the addresses of the axes to be traversed.

Commands REPOSA, REPOSQA and REPOSHA automatically reposition all axes. Individual axis names need not be specified.
When commands REPOSL, REPOSQ and REPOSH are programmed, all geometry axes are traversed automatically, i.e. they need not be named in the command. All other axes to be repositioned must be specified in the commands.

Approach along a straight line, REPOSA, REPOSL
The tool approaches the repositioning point along a straight line.

All axes are automatically traversed with command REPOSA. With REPOSL you can specify which axes are to be moved.

Example:
REPOSL RMI DISPR=6 F400
or
REPOSA RMI DISPR=6 F400

Approach along quadrant, REPOSQ, REPOSQA
The tool approaches the repositioning point along a quadrant with a radius of DISR=... The control system automatically calculates the intermediate point between the start and repositioning points.

Example:
REPOSQ RMI DISR=10 F400
Approach along semi-circle, REPOSH, REPOSQA
The tool approaches the repositioning point along a semi-circle with a diameter of DISR=.... The control system automatically calculates the intermediate point between the start and repositioning points.

Example:
REPOSH RMI DISR=20 F400

The following applies to circular motions REPOSH and REPOSQ:
The circle is traversed in the specified working planes G17 to G19.
If you specify the third geometry axis (infeed direction) in the approach block, the repositioning point is approached along a helix in case the tool position and programmed position in the infeed direction do not coincide.

In the following cases, the control automatically switches over to linear approach REPOSQ:
- You have not specified a value for DISR.
- No defined approach direction is available (program interruption in a block without travel information).
- With an approach direction that is perpendicular to the current working plane.
Motion-Synchronous Action

10.1 Structure, basic information ........................................................................................................... 10-395
10.1.1 Programming and command elements .......................................................................................... 10-397
10.1.2 Validity range: Identification number ID ..................................................................................... 10-398
10.1.3 Vocabulary word ............................................................................................................................ 10-399
10.1.4 Actions .......................................................................................................................................... 10-402
10.1.5 Overview of synchronized actions .................................................................................................. 10-404

10.2 Basic modules for conditions and actions ......................................................................................... 10-406

10.3 Special real-time variables for synchronized actions .......................................................................... 10-409
10.3.1 Flags/counters $AC_MARKER[n]$ .................................................................................................. 10-409
10.3.2 Timer variable $AC_TIMER[n]$, SW 4 and higher ......................................................................... 10-409
10.3.3 Synchronized action parameters $AC_PARAM[n]$ ........................................................................ 10-410
10.3.4 Access to R parameters $Rxx$ ....................................................................................................... 10-411
10.3.5 Machine and setting data read/write (SW 4 and higher) .............................................................. 10-412
10.3.6 FIFO variable $AC_FIFO1[n] ... $AC_FIFO10[n]$ (SW 4 and higher) ........................................ 10-413

10.4 Actions within synchronized actions ............................................................................................... 10-415
10.4.1 Auxiliary functions output ............................................................................................................. 10-415
10.4.2 Set read-in disable RDISABLE ....................................................................................................... 10-416
10.4.3 Cancel preprocessing stop STOPREOF ......................................................................................... 10-417
10.4.4 Deletion of distance-to-go ............................................................................................................. 10-418
10.4.5 Delete distance-to-go with preparation, DELDTG, DELDTG ("Axis 1 to x") ................................ 10-418
10.4.6 Polynomial definition, FCTDEF, block-synchronized ................................................................. 10-420
10.4.7 Laser power control ....................................................................................................................... 10-422
10.4.8 Evaluation function SYNFCT ...................................................................................................... 10-423
10.4.9 Adaptive control (additive) ........................................................................................................... 10-424
10.4.10 Adaptive control (multiplicative) ................................................................................................. 10-425
10.4.11 Clearance control with limited compensation .............................................................................. 10-426
10.4.12 Online tool offset FTOC ............................................................................................................. 10-428
10.4.13 Positioning movements ................................................................................................................ 10-430
10.4.14 Position axis POS .......................................................................................................................... 10-432
10.4.15 Start/stop axis MOV ..................................................................................................................... 10-432
10.4.16 Axial feed FA ................................................................................................................................. 10-433
10.4.17 SW limit switch ............................................................................................................................... 10-434
10.4.18 Axis coordination ........................................................................................................................... 10-434
10.4.19 Set actual value .............................................................................................................................. 10-436
10.4.20 Spindle motions ............................................................................................................................. 10-437
10.4.21 Coupled-axis motion TRAILON, TRAILOF ............................................................................. 10-438
10.4.22 Leading value coupling LEADON, LEADOF ............................................................................ 10-439
10.4.23 Measurement ................................................................................................................................. 10-441
10.4.24 Set/clear wait marks: SETM, CLEARM (SW 5.2 and higher) .................................................. 10-441
10.4.25 Error responses

10.4.26 Travel to fixed stop FXS and FOCON/FOCOF

10.5 Technology cycles

10.5.1 Lock, unlock, reset: LOCK, UNLOCK, RESET

10.6 Cancel synchronized action: CANCEL

10.7 Supplementary conditions
10.1 Structure, basic information

Function

Synchronized actions allow you to start different actions from the current parts program and to execute them synchronously.

The starting point of these actions can be defined with conditions evaluated in real time (in interpolation cycles). The actions are therefore responses to real-time events, execution of them is not limited by block boundaries.

A synchronized action also contains information about the effectiveness of the actions and about the frequency with which the programmed real-time variables are scanned and therefore about the frequency with which the actions are started. In this way, an action can be triggered just once or cyclically in interpolation cycles.

Programming

<table>
<thead>
<tr>
<th>Identification number ID/IDS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID=n</strong></td>
</tr>
<tr>
<td>Modal synchronized actions in automatic mode, local to program; n = 1...255</td>
</tr>
<tr>
<td><strong>IDS=n</strong></td>
</tr>
<tr>
<td>Modal synchronized actions in each mode, static; n = 1...255</td>
</tr>
<tr>
<td><strong>Without ID/IDS</strong></td>
</tr>
<tr>
<td>Non-modal synchronized actions in automatic mode</td>
</tr>
<tr>
<td><strong>Vocabulary word</strong></td>
</tr>
<tr>
<td>Execution of the action is not subject to any condition. The action is executed cyclically in any interpolation cycles.</td>
</tr>
</tbody>
</table>

Explanation

DO Action1 Action2 ... 
VOCABULARY_WORD condition DO action1 action2 ...
ID=n VOCABULARY_WORD condition DO action1 action2 ...
IDS=n VOCABULARY_WORD condition DO action1 action2 ...
### 10Motion-Synchronous Action

#### 10.1 Structure, basic information

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN</td>
<td>The condition is tested until it is fulfilled once, the associated action is executed once.</td>
</tr>
<tr>
<td>WHENEVER</td>
<td>The condition is tested cyclically. The associated action is executed cyclically while the condition is fulfilled.</td>
</tr>
<tr>
<td>FROM</td>
<td>After the condition has been fulfilled once, the action is executed cyclically while the synchronized action is active.</td>
</tr>
<tr>
<td>EVERY</td>
<td>The action is initiated once when the condition is fulfilled and is executed again when the condition changes from the FALSE state to the TRUE state. The condition is tested cyclically. Every time the condition is fulfilled, the associated action is executed.</td>
</tr>
</tbody>
</table>

#### Condition

Gating logic for real-time variables, the conditions are checked in the interpolation cycle.

In SW 5 and higher, the G codes can be programmed in synchronized actions for condition evaluation.

#### DO

Triggers the action if the condition is fulfilled.

#### Action

Action started if the condition is fulfilled, e.g. assign variable, activate axis coupling, set NCK outputs, output M and H functions, ... In SW 5 and higher, the G codes can be programmed in synchronized actions for actions/technology cycles.

#### Coordination of synchronized actions/technology cycles

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCEL[n]</td>
<td>Cancel synchronized action</td>
</tr>
<tr>
<td>LOCK[n]</td>
<td>Inhibit technology cycle</td>
</tr>
<tr>
<td>UNLOCK[n]</td>
<td>Enable technology cycle</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset technology cycle</td>
</tr>
</tbody>
</table>

#### Programming example

```
WHEN $AA_IW[Q1]>5 DO M172 H510  ;If the actual value of axis Q1 exceeds 5 mm, auxiliary functions M172 and H510 are output to the PLC interface.
```

If real-time variables occur in a parts program (e.g. actual value, position of a digital input or output etc.), preprocessing is stopped until the previous block has been executed and the values of the real-time variables obtained. The real-time variables used are evaluated in interpolation cycles.

Advantages with synchronized actions:

- Preprocessing is not stopped.
10.1 Structure, basic information

Possible applications:
- Optimization of runtime-critical applications (e.g. tool changing)
- Fast response to an external event
- Programming AC controls
- Setting up safety functions
- ....

10.1.1 Programming and command elements

Function

A synchronized action is programmed on its own in a separate block and triggers a machine function in the next executable block (e.g. traversing movement with G0, G1, G2, G3; block with auxiliary function output).

Synchronized actions consist of up to five command elements each with a different task:

Example:

| ID=1 | WHENEVER | $A\_IN[1]==1 | DO | $A\_OUT[1]=1 |

Synchronized action no. 1: whenever input 1 is set then set output 1
10.1.2 Validity range: Identification number ID

Function

The scope of validity of a synchronized action is defined by the identification number (modal ID):

- **No modal ID**
  
The synchronized action is active in automatic mode only. It applies only to the next executable block (block with motion instructions or other machine action), is **non-modal**.
  
  **Example:**
  ```
  G1 X20 ;Executable block
  ```

- **ID=n; n=1...255**
  
The synchronized action applies **modally** in the following blocks and is deactivated by CANCEL(n) or by programming a new synchronized action with the same ID. The synchronized actions that apply in the M30 block are also still active (if necessary deactivate with the CANCEL command).
  
  ID synchronized actions only apply in **automatic** mode.
  
  **Example:**
  ```
  ID=2 EVERY $A\_IN[1]==1 DO POS[X]=0
  ```

- **IDS=n; n=1...255**
  
  These **static** synchronized actions apply **modally** in **all operating modes**.
  
  They can be defined not only for starting from a parts program but also directly after power-on from an asynchronous subprogram (ASUB) started by the PLC. In this way, actions can be activated that are executed regardless of the mode selected in the NC.
  
  **Example:**
  ```
  IDS=1 EVERY $A\_IN[1]==1 DO POS[X]=100
  ```
10.1 Structure, basic information

Application:
- AC loops in JOG mode
- Logic operations for Safety Integrated
- Monitoring functions, responses to machine states in all modes

Sequence of execution
Synchronized actions that apply modally or statically are executed in the order of their ID(S) numbers (in the interpolation cycle).
Non-modal synchronized actions (without ID number) are executed in the programmed sequence after execution of the modal synchronized actions.

10.1.3 Vocabulary word

Function
The vocabulary word determines how many times the following condition is to be scanned and the associated action executed.
- No vocabulary word:
  If no vocabulary word is programmed, the condition is considered to be always fulfilled. The synchronous commands are executed cyclically.
- WHEN
  The condition is scanned in each interpolation cycle until it is fulfilled once, whereupon the action is executed once.
- WHenever
  The condition is scanned in each interpolation cycle. The action is executed in each interpolation cycle while the condition is fulfilled.

Example:
```
DO $A_OUTA[1]=$AA_IN[X]
;Output of actual value on analog output
```
Motion-Synchronous Action

10.1 Structure, basic information

- FROM
  The condition is tested in each interpolation cycle until it is fulfilled once. The action is then executed while the synchronous action is active, i.e. even if the condition is no longer fulfilled.

- EVERY
  The condition is scanned in each interpolation cycle. The action is executed once whenever the condition is fulfilled.
  Pulse edge control:
  The action is initiated again when the condition changes from FALSE to TRUE.

Condition
Defines whether an action is to be executed by comparing two real-time variables or one real-time variable with an expression calculated during preprocessing.

SW 4 and higher:
Results of comparisons can also be gated by Boolean operators in the condition ().
The condition is tested in interpolation cycles. If it is fulfilled, the associated action is executed.

SW 5 and higher:
Conditions can be specified with a G code. This means that it is possible to have defined settings for condition evaluation and the action/technology cycle irrespective of the currently active parts program state. It is necessary to decouple synchronized actions from the programming environment because synchronized actions are to execute their actions in the defined initial state at any time when the trigger conditions are fulfilled.

Application cases:
Defining the measurement systems for condition assessment and action via G codes G70, G71, G700, G710.

Example:
ID=1 EVERY $AA.IM[B]>75 DO
POS[U]=IC(10) FA[U]=900;
When the actual value of axis B overshoots the value 75 in machine coordinates, the U axis should move forwards by 10 with an axial feed.
In SW 5 only these G codes are allowed.
A specified G code for the condition applies for
assessment of the condition as well as for the
action if there is no separate G code specified for the
action.
Only one G code of the G code group may be
programmed for each condition part.

Programming example

| WHENEVER $AA_IM[X] > 10.5*SIN(45) DO ... | Comparison with an expression calculated during preprocessing |
| WHENEVER $AA_IM[X] > $AA_IM[X1] DO ... | Comparison with other real-time variable |
| WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ... | Two logic-gated comparisons |

Possible conditions:
- Comparison of real-time variables
  (analog/digital inputs/outputs, etc.)
- Boolean gating of comparison results
- Computation of real-time expressions
- Time/distance from beginning of block
- Distance from block end
- Measured values, measured results
- Servo values
- Velocities, axis status
10.1.4 Actions

Function

In each synchronized action, you can program one or more actions. All actions programmed in a block are started in the same interpolation cycle.

In SW 5 and higher, actions can be used with a G code for the action/technology cycle. This G code specifies another G code from the one set for the condition for all actions in the block and technology cycles if necessary. If there are technology cycles in the action part, then after completion of the technology cycle the G code continues to apply modally for all subsequent actions until the next G code.

Only a G code from the G code group (G70, G71, G700, G710) may be programmed.

Possible actions:
- Assign variables
- Write setting data
- Set control parameters
- DELDTG: Delete fast distance-to-go
- RDISABLE: Set read-in disable
- Output M, S and H auxiliary functions
- STOPREOF: Cancel preprocessing stop
- FTOC: Online tool offset
- Definition of evaluation functions (polynomials)
- SYNFCT: Activate evaluation functions: AC control
- Switchover between several feedrates in a programmed block depending on binary and analog signals
- Feedrate overrides
- Start/position/stop positioning axes (POS) and spindles (SPOS)
- PRESETON: Set actual value
10.2 Motion-Synchronous Action

10.1 Structure, basic information

- Activate or deactivate coupled-axis motion/leading value coupling
- Measurement
- Set up additional safety functions
- Output of digital and analog signals
- ...

Programming example

Synchronized action with two actions

WHEN $AA_IM[Y] >= 35.7 DO M135 $AC_PARAM=50

If the condition is fulfilled, M135 is output to the PLC and the override is set to 50%.

As the action, you can also specify a program (single-axis program, technology cycle). This must only comprise those actions that can also be programmed individually in synchronized actions. The individual actions of such a program are executed sequentially in interpolation cycles.

Note

Actions can be executed whatever mode is selected. The following actions are only active in automatic mode when the program is active
- STOPREOF,
- DELDTG.
10.1.5 Overview of synchronized actions

**SW 3.x and lower**
- Programming of sequences in the interpolation cycle at the user level (parts program)
- Response to events/statuses in the interpolation cycle
- Gating logic in real time
- Access to I/Os, control status and machine status
- Programming of cyclic sequences that are executed in the interpolation cycle
- Triggering of specific NC functions (read-in disable, axially overlaid motion, ...)
- Execution of technology functions in parallel with path motion
- Triggering of technology functions regardless of block boundaries

**SW 4 and higher**
- Diagnosis possible for synchronized actions
- Expansion of the main run variable used in synchronized actions
- Complex conditions in synchronized actions
- Expansion of expressions in synchronized actions: Combination of real-time variables with basic arithmetic operations and functions in the interpolation cycle, indirect addressing of main run variables via index can be changed online
- Setting data from synchronized actions can be modified and evaluated online
- Configuration possibilities: Number of simultaneously active synchronized actions can be set via machine data.
- Start positioning axis motion and spindles from synchronized actions (command axes)
- Preset from synchronized actions
- Activation, deactivation, parameterization of axis coupling: Leading value coupling, coupled-axis motion
- Activation/deactivation of axial measuring function
- Software cams
11.02 Motion-Synchronous Action

10.1 Structure, basic information

- Delete distance-to-go without stopping preprocessing
- Single-axis programs, technology cycles
- Synchronized actions active in JOG mode beyond the boundaries of the program
- Synchronized actions that can be influenced from the PLC
- Protected synchronized actions
- Expansion for overlaid motion / clearance control

**SW 5.x and higher**

- Travel to fixed stop FXS:
  Synchronized actions, triggered with FXS, FXST and FXSW
- Travel with limited moment/force FOC:
  Synchronized action is activated either modally or non-modally with FOCON and deactivated with FOCOF.
10.2 Basic modules for conditions and actions

Real-time variables

Real-time variables are evaluated and written in the interpolation cycle.

The real-time variables are:
- $A...$, main run variable,
- $V...$, servo variable.

To identify them specially, these variables can be programmed with $\$: $AA_{IM}[X]$ is equivalent to $AA_{IM}[X]$.

Setting and machine data must be identified with $\$ when evaluation/assignment takes place in the interpolation cycle.

A list of variables is given in the Appendix.

Calculations in real time

Calculations in real time are restricted to the data types INT, REAL and BOOL.

Real-time expressions are calculations that can be executed in interpolation cycles that can be used in the condition and the action for assignment to NC addresses and variables.

- **Comparisons**
  In conditions, variables or partial expressions of the same data type can be compared. The result is always of data type BOOL.
  All the usual comparison operators are permissible (==, <>, <, >, <=, >=).

- **Boolean operators**
  Variables, constants and comparisons can be gated using the usual Boolean operators (NOT, AND, OR, XOR)
• **Bit operators**
  The bit operators B_NOT, B_AND, B_OR, B_XOR can be used. Operands are variables or constants of the INTEGER type.

• **Basic arithmetic operations**
  Real-time variables of types INTEGER and REAL can be subjected to the basic arithmetic operations, with each other or with a constant (+, -, *, /, DIV, MOD).

• **Mathematical functions**
  Mathematical functions cannot be applied to real-time variables of data type REAL (SIN, COS, TAN, ASIN, ACOS, ABS, TRUNC, ROUND, LN, EXP, ATAN2, ATAN, POT, SQRT, CTAB, CTABINV).

  **Example:**
  
  ```
  DO $AC_PARAM[3] = COS($AC_PARAM[1])
  ```
Notes
Only variables of the same data type can be gated.

Correct: $R10=$AC_PARAM[1]
Incorrect: $R10=$AC_MARKER[1]

Multiplication and division are performed before addition and subtraction and bracketing of expressions is permissible.

The operators DIV and MOD are permissible for the data type REAL (SW 4 and higher).

Example:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHENEVER $AA_IM[x2] &lt; $AA_IM[x1] - 1.9 DO $A_OUT[5] = 1</td>
<td>Subtraction of a constant from real-time variable</td>
</tr>
</tbody>
</table>

Indexation
Real-time variables can be indexed with real-time variables.

Notes
Variables that are not formed in real time must not be indexed with real-time variables.

Example:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN... DO $AC_PARAM[$AC_MARKER[1]] = 3</td>
<td></td>
</tr>
</tbody>
</table>

Illegal:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
</table>

Programming example
Example of real-time expressions

<table>
<thead>
<tr>
<th>ID=1 WHENEVER ($AA_IM[Y]&gt;30) AND ($AA_IM[Y]&lt;40) DO $AA_OVR[S1]=80</th>
<th>Selection of a position window</th>
</tr>
</thead>
</table>
10.3 Special real-time variables for synchronized actions

The real-time variables listed below can be used in synchronized actions:

10.3.1 Flags/counters $AC_MARKER[n]

Function

Flag variables can be read and written in synchronized actions.

Channel-specific flags/counters

$AC_MARKER[n]

Data type: INTEGER

A channel-specific flag variable exists under the same name once in each channel.

Example:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN ... DO $AC_MARKER[0] = 2</td>
<td></td>
</tr>
<tr>
<td>WHEN ... DO $AC_MARKER[0] = 3</td>
<td></td>
</tr>
<tr>
<td>WHEN $AC_MARKER == 3 DO $AC_OVR=50</td>
<td></td>
</tr>
</tbody>
</table>

10.3.2 Timer variable $AC_TIMER[n], SW 4 and higher

Function

(not 840D NCU 571, FM-NC)

The system variable $AC_TIMER[n] allows actions to be started following defined waiting times.

Data type: REAL

Units: s

n: Number of the timer variable

- **Set timer**

  A timer variable is incremented via value assignment $AC_TIMER[n]=value
10.3 Special real-time variables for synchronized actions

- **Halt timer**
  Incrementation of a timer variable is halted by assigning a negative value $AC_TIMER[n]=-1$

- **Read timer**
  The current time value can be read when the timer is running or when it has stopped. When the timer is stopped by assigning the value -1, the most up-to-date timer value is retained and can be read.

**Example:**
Output of an actual value via analog output
500 ms after detection of a digital input

```plaintext
WHEN $A_IN[1] == 1 DO $AC_TIMER[1]=0 ; Reset and start timer
```

10.3.3 Synchronized action parameters $AC_PARAM[n]$

**Function**

Data type: REAL

$n$: Number of parameter 0-n

Synchronized action parameters $AC_PARAM[n]$ are used for calculations and as a buffer in the synchronized actions.

The number of available AC parameter variables per channel are defined using machine data MD 28254: MM_NUM_AC_PARAM.

The parameters are available once per channel under the same name. The $AC_PARAM$ flags are stored in the dynamic memory.
10.3 Special real-time variables for synchronized actions

10.3.4 Access to R parameters $Rxx

**Function**

Data type: REAL

These static variables are used for calculations in the parts program etc. They can be addressed in the interpolation cycle by appending $.

**Examples:**

| WHEN $AA\_IM\[X\] >= 40.5 DO $R10 = $AA\_MM\[Y\] | Write access to the R parameter 10. |
| WHEN $AA\_IM\[X\] >= 6.7 DO $R[$AC\_MARKER\[1\]\] = 30.6; Read access to the R parameter whose number is given in flag 1 |

**Notes**

**Application:**

The use of R parameters in synchronized actions permits

- storage of values that you want to retain beyond the end of program, NC reset, and Power On.
- display of stored value in the R parameter display
- archiving of values determined for synchronized actions

The R parameters must be used either as "normal" arithmetic variables Rxx or as real-time variables $Rxx.

If you want the R parameter to be used as a "normal" arithmetic variable again after it has been used in a synchronized action, make sure that the preprocessing stop is programmed explicitly with STOPRE for synchronization of preprocessing and the main run:

**Example:**

| WHEN $AA\_IM\[X\] >= 40.5 DO $R10 = $AA\_MM\[Y\] | Use of R10 in synchronized actions |
| G01 X500 Y70 F1000 | Preprocessing stop |
| STOPRE | Evaluation of the arithmetic variable |

© Siemens AG, 2002. All rights reserved
10.3 Special real-time variable for synchronized actions

10.3.5 Machine and setting data read/write (SW 4 and higher)

Function

From SW 4 and higher, it is possible to read and write the machine and setting data (MD, SD) of synchronized actions.

- **Read fixed MD, SD**
  They are addressed from within the synchronized action in the same manner as in normal parts program commands and are preceded by a $ character.
  **Example:**
  ```
  ID=2 WHENEVER $AA_IM[z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
  
  ;In this example, reverse position 2 for oscillation is addressed assumed to be unmodifiable.
  ```

- **Read modifiable MD, SD**
  They are addressed from within the synchronized action, preceded by $$ characters and evaluated in the interpolation cycle.
  **Example:**
  ```
  ID=1 WHENEVER $AA_IM[z]<$$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
  
  ;It is assumed here that the reverse position can be modified by a command during machining.
  ```

- **Write MD, SD**
  **Precondition:**
  The current setting for access authorization must permit write access. It is only appropriate to modify MD and SD from the synchronized action when the change is active immediately. The active states are listed for all MD and SD in
  **References:** /LIS/, Lists
  **Addressing:**
  The MD and SD to be modified must be addressed preceded by $$.
  **Example:**
  ```
  ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]=20
  $$SN_SW_CAM_MINUS_POS_TAB_1[0]=30
  
  ;Changing the switching position of SW cams. Note: The switching positions must be changed two to three interpolation cycles before they reach their position.
  ```
10.3.6 FIFO variable $AC\_FIFO1[n] \ldots $AC\_FIFO10[n] (SW 4 and higher)

**Function**

Data type: REAL
10 FIFO variables (circulating buffer store) are available to store associated data sequences.
Application:
- Cyclic measurement
- Pass execution

Each element can be accessed in read or write mode.
The number of available FIFO variables is defined using machine data MD 28260: NUM\_AC\_FIFO.
The number of values that can be written into an FIFO variable is defined using the machine data MD 28264: LEN\_AC\_FIFO. All FIFO variables are of the same length.

Indices 0 to 5 have a special significance:
n=0: While writing: New value is stored in FIFO
    While reading: Oldest element is read and removed from FIFO
n=1: Accessing the oldest stored element
n=2: Accessing the most recently stored element
n=3: Sum of all FIFO elements
n=4: Number of elements available in FIFO.
    Read and write access to each element is possible.
    FIFO variables are reset by resetting the number of elements, e.g. for the first FIFO variable: $AC\_FIFO1[4]=0
n=5: Current write index relative to start of FIFO
n=6 to 6+nmax: Access to nth FIFO element
Programming example

Circulating memory

During a production run, a conveyor belt is used to transport products of different lengths (a, b, c, d). The conveyor belt of transport length "I" therefore carries a varying number of products depending on the lengths of individual products involved in the process. With a constant speed of transport, the function for removing the products from the belt must be adapted to the variable arrival times of the products.

DEF REAL INTV=2.5
Constant distance between products placed on the belt.

DEF REAL TOTAL=270
Distance between length measurement and removal position.

Reset FIFO at beginning of process.

EVERY $A_IN[2]==1 DO $AC_TIMER[0]=0
If a product interrupts the light barrier, start timing.

EVERY $A_IN[2]==0 DO $AC_FIFO1[0]=$AC_TIMER[0]*$AA_VACTM[B]
If the light barrier is free, calculate and store in the FIFO the product length from the time measured and the velocity of transport.

$R1=$AC_FIFO1[0]
As soon as the sum of all product lengths and intervals between products is greater than or equal to the length between the placement and the removal position, remove the product from the conveyor belt at the removal position, read the product length out of the FIFO.
10.4 Actions within synchronized actions

10.4.1 Auxiliary functions output

**Function**
If the conditions are fulfilled, up to 10 M, H and S functions can be output per machining block.
Auxiliary function output is activated using the action codeword "DO".
The auxiliary functions are output immediately in the interpolation cycle. The output timing defined in the machine data for auxiliary functions is not active.
The output timing is determined when the condition is fulfilled.

**Example:**
Switch on coolant at a specific axis position:
WHEN $AA\_IM[X] >= 15 DO M07
POS[X] = 20 FA[X] = 250

**Sequence**
Auxiliary functions must only be programmed with the vocabulary words WHEN or EVERY in non-modal synchronized actions (without model ID).
Whether an auxiliary function is active or not is determined by the PLC, e.g. via NC start.

**Notes**
Not possible from a motion synchronized action:
- M0, M1, M2, M17, M30: Program halt/end (M2, M17, M30 possible for technology cycle)
- M70: Spindle function
- M functions for tool change set with M6 or via machine data
- M40, M41, M42, M43, M44, M45: Gear change

**Programming example**
WHEN $AA\_IW[Q1] > 5 DO M172 H510
If the actual value of axis Q1 exceeds 5 mm, auxiliary functions M172 and H510 are output to the PLC.
10.4 Actions within synchronized actions

10.4.2 Set read-in disable RDISABLE

Function

With RDISABLE further block execution is stopped in the main program if the condition is fulfilled. Programmed synchronized motion actions are still executed, the following blocks are still prepared.

At the beginning of the block with RDISABLE, exact positioning is always triggered whether RDISABLE is active or not.

Programming example

Start the program in interpolation cycles dependent on external inputs.

... WHENEVER $A_INA[2]<7000 DO RDISABLE ;If the voltage 7V is exceeded at input 2, the program is stopped (1000= 1V).
N10 G1 X10 ;When the condition is fulfilled, the read-in disable is active at the end of N10
N20 G1 X10 Y20
...
...
10.4.3 Cancel preprocessing stop STOPREOF

**Function**

In the case of an explicitly programmed preprocessing stop STOPRE or a preprocessing stop implicitly activated by an active synchronized action, STOPREOF cancels the preprocessing stop after the next machining block as soon as the condition is fulfilled.

**Notes**

STOPREOF must be programmed with the vocabulary word WHEN and non modally (without ID number).

**Programming example**

Fast program branch at the end of the block.

```
WHEN $AC_DTEB<5 DO STOPREOF ;Cancel the preprocess stop when distance to block end is less than 5mm.
G01 X100 ;The preprocessing stop is canceled after execution of the linear interpolation.
IF $A_INA[7]>500 GOTOF MARKE1=X100 ;If the voltage 5V is exceeded at input 7, jump to label 1.
```
10.4 Actions within synchronized actions

10.4.4 Deletion of distance-to-go

Delete distance-to-go can be triggered for a path and for specified axes depending on a condition.

The possibilities are:
- Fast, prepared delete distance-to-go
- Delete distance-to-go without preparation (SW 4.3 and higher)

10.4.5 Delete distance-to-go with preparation, DELDTG, DELDTG ("Axis 1 to x")

Notes
- The axis designation contained in brackets behind DELDTG is only valid for one positioning axis.

Function
- Prepared delete distance-to-go with DELDTG permits a fast response to the triggering event and is therefore used for time-critical applications, e.g., if
  - the time between delete distance-to-go and the start of the next block must be very short.
  - the condition for delete distance-to-go will very probably be fulfilled.

Sequence
- At the end of a traversing block in which a prepared delete distance-to-go was triggered, preprocess stop is activated implicitly.
- Continuous path mode or positioning axis movements are therefore interrupted or stopped at the end of the block with fast delete distance-to-go.
10.4 Actions within synchronized actions

Programming example

Rapid delete distance-to-go path

```
WHEN $A_IN[1]==1 DO DELDTG
N100 G01 X100 Y100 F1000 ;When the input is set, the movement is canceled
N110 G01 X...
IF $AA_DELT>50...
```

Programming example

Rapid axial delete distance-to-go

Stopping a programmed positioning movement:

```
ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700  Start axis
WHEN $A_IN[2]==1 DO DELDTG(V)  Delete distance-to-go, the axis is stopped using MOV=0
```

Delete distance-to-go depending on the input voltage:

```
WHEN $A_INA[5]>8000 DO DELDTG(X1)
;As soon as voltage on input 5 exceeds 8V, delete distance-to-go for axis X1.
POS[X1]=100 FA[X1]=10 G1 Z100 F1000
```

Restriction

Prepared delete distance-to-go

- cannot be used with active tool radius correction.
- the action must only be programmed in non-modal synchronized actions (without ID number).
10.4.6 Polynomial definition, FCTDEF, block-synchronized

Programming

FCTDEF(Polynomial_No., LLIMIT, ULIMIT, a_0, a_1, a_2, a_3)

Explanation

<table>
<thead>
<tr>
<th>Polynomial_No.</th>
<th>Number of the 3rd degree polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLIMIT</td>
<td>Lower limit for function value</td>
</tr>
<tr>
<td>ULIMIT</td>
<td>Upper limit for function value</td>
</tr>
<tr>
<td>a_0, a_1, a_2, a_3</td>
<td>Polynomial coefficient</td>
</tr>
</tbody>
</table>

Function

FCTDEF allows 3rd degree polynomials to be defined as \( y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \). These polynomials are used by the online tool offset FTOC and the evaluation function SYNFCFT to calculate function values from the main run variables (real-time variables).

The polynomials are defined either block-synchronized with the function FCTDEF or via system variables:

<table>
<thead>
<tr>
<th>$\text{AC}_\text{FCTLL}[n]$</th>
<th>Lower limit for function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC}_\text{FCTUL}[n]$</td>
<td>Upper limit for function value</td>
</tr>
<tr>
<td>$\text{AC}_\text{FCT0}[n]$</td>
<td>( a_0 )</td>
</tr>
<tr>
<td>$\text{AC}_\text{FCT1}[n]$</td>
<td>( a_1 )</td>
</tr>
<tr>
<td>$\text{AC}_\text{FCT2}[n]$</td>
<td>( a_2 )</td>
</tr>
<tr>
<td>$\text{AC}_\text{FCT3}[n]$</td>
<td>( a_3 )</td>
</tr>
<tr>
<td>( n )</td>
<td>Number of the polynomial</td>
</tr>
</tbody>
</table>
Notes

- The system variables can be written from the parts program or from a synchronized action. When writing from parts programs, program STOPRE to ensure that writing is block synchronized.
- SW 4 and higher:
  The system variables $AC_FCTLL[n]$, $AC_FCTUL[n]$, $AC_FCT0[n]$ to $AC_FCTn[n]$ can be modified from within synchronized actions (not SINUMERIK FM-NC, not SINUMERIK 840D with NCU 571).

When writing from synchronized actions, the polynomial coefficients and function value limits are active immediately.

Programming example

Polynomial for straight section:

With upper limit 1000, lower limit -1000, ordinate section $a_0=\$AA\_IM[X]$ and linear gradient 1 the polynomial is:

\[
FCTDEF(1, -1000, 1000, \$AA\_IM[X], 1)
\]
10.4.7 Laser power control

Programming example

Polynomial definition using variables

One of the possible applications of polynomial definition is the laser output control. Laser output control means:
Influencing the analog output in dependence on, for example, the path velocity.

```
$AC_FCTLL[1]=0.2  \quad \text{Definition of the polynomial coefficient}
$AC_FCTUL[1]=0.5
$AC_FCT0[1]=0.35
$AC_FCT1[1]=1.5EX-5
STOPRE
ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1 +0.35  \quad \text{Changing the upper limit online.}
ID=2 DO SYNFCT(1,$A_OUTA[1],$AC_VACTW)
```

In dependence on the path velocity (stored in $AC_VACTW) the laser output control is controlled via analog output 1.

Note

The polynomial defined above is used with SYNFCT.
10.4.8 Evaluation function SYNFCF

Programming

SYNFCF(Polynomial_No., realtime variable output, real-time variable input)

Explanation

<table>
<thead>
<tr>
<th>Polynomial_No.</th>
<th>With polynomial defined with FCTDEF (see Subsection &quot;Polynomial definition&quot;).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-time variable output</td>
<td>Write real-time variable</td>
</tr>
<tr>
<td>Real-time variable input</td>
<td>Read real-time variable</td>
</tr>
</tbody>
</table>

Function

SYNFCF reads real-time variables in synchronism with execution (e.g. analog input, actual value, ...) and uses them to calculate function values up to the 3rd degree (e.g. override, velocity, axis position, ...) using an evaluation polynomial (FCTDEF). The result is output in to real-time variables and subjected to upper and lower limits with FCTDEF (see Subsection 10.4.7).

As real-time variables, variables can be selected and directly included in the processing operation
- with additive influencing
- with multiplicative influencing
- as position offset
- directly.

Application

The evaluation function is used
- in AC control (Adaptive Control)
- in laser output control
- with position feedforward
10.4 Actions within synchronized actions

10.4.9 Adaptive control (additive)

Programming example

Additive influence on the programmed feedrate

A programmed feedrate is to be controlled by adding using the current of the X axis (infeed axis): The feedrate should only vary by +/- 100 mm/min and the current fluctuates by +/-1A around the working point of 5A.

1. Polynomial definition
   Determination of the coefficients
   \[ y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 \]
   \[ a_1 = -100 \text{mm/min A} \]
   \[ a_0 = -(100)*5 = 500 \]
   \[ a_2 = a_3 = 0 \] (not quadratic or cubic element)
   Upper limit = 100
   Lower limit = -100

   Therefore:
   \[ \text{FCTDEF}(1, -100, 100, 500, -100, 0, 0) \]

2. Activate AC control
   \[ \text{ID=1 DO SYNFCT}(1, \$AC_{VC}, \$AA_{LOAD}[x]) \]
   ;Read the current axis load (% of the max. drive current) via \$AA_{LOAD}[x],
   calculate the path feedrate override with the polynomial defined above.
10.4.10 Adaptive control (multiplicative)

Programming example

Influence the programmed feedrate by multiplication

The aim is to influence the programmed feedrate by multiplication. The feedrate must not exceed certain limits – depending on the load on the drive:

- The feedrate is to be stopped at a drive load of 80%: Override = 0.
- At a drive load of 30% it is possible to traverse at programmed feedrate: Override = 100%.
- The feedrate can be exceeded by 20%: Max. override = 120%.

1. Polynomial definition

Determination of the coefficients

\[ y = f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \]

- \[ a_1 = \frac{-100\%/(80-30)\%}{2} = -2 \]
- \[ a_0 = 100 + (2 \times 30) = 160 \]
- \[ a_2 = a_3 = 0 \] (not quadratic or cubic element)

Upper limit = 120
Lower limit = 0

Therefore:

\[ \text{FCTDEF}(2,0,120,160,-2,0,0) \]

2. Activate AC control

\[ \text{ID}=1 \ \text{DO \ SYNFCT}(2,\$AC\_OVR,\$AA\_LOAD[x]) \]

:Read the current axis load (% of the max. drive current) via \$AA\_LOAD[x],
calculate the feedrate override with the polynomial defined above.
10.4.11 Clearance control with limited compensation

Programming example

Integrating calculation of the distance values with boundary check

$\text{AA\_OFF\_MODE} = 1$

**Important:**
The loop gain of the overlying control loop depends on the setting for the interpolation cycle.
Remedy: Read MD for interpolation cycle and take it into account.

**Note:**
Restriction of the velocity of the overlying interpolator with MD 32020: $\text{JOG\_VELO}$
with an interpolation cycle of 12 ms:

Velocity:

\[
\frac{0.120 \text{mm}}{0.6 \text{ms}} = 0.6 \text{m/min}
\]

Subroutine: Clearance control ON

<table>
<thead>
<tr>
<th>%_N_AON_SPF</th>
<th>Subroutine for clearance control ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AA_OFF_LIMIT}[Z]=1$</td>
<td>Determine limiting value</td>
</tr>
<tr>
<td>FCTDEF(1, -10, +10, 0, 0.6, 0.12)</td>
<td>Polynomial definition</td>
</tr>
<tr>
<td>ID=1 DO SYNFCF(1, $\text{AA_OFF}[Z], \text{A_INA}[3])</td>
<td>Clearance control active</td>
</tr>
<tr>
<td>ID=2 WHENEVER $\text{AA_OFF_LIMIT}[Z]&lt;&gt;0$ DO $\text{AA_OVR}[X] = 0$</td>
<td>Disable axis $X$ when limit value is overshot</td>
</tr>
<tr>
<td>RET</td>
<td>ENDPROC</td>
</tr>
</tbody>
</table>

Subroutine: Clearance control OFF

<table>
<thead>
<tr>
<th>%_N_AOFF_SPF</th>
<th>Subroutine for clearance control OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC AOFF</td>
<td>Cancel clearance control synchronized action</td>
</tr>
<tr>
<td>CANCEL(1)</td>
<td>Cancel clearance control synchronized action</td>
</tr>
<tr>
<td>CANCEL(2)</td>
<td>Cancel off limits check</td>
</tr>
<tr>
<td>RET</td>
<td>ENDPROC</td>
</tr>
</tbody>
</table>
Main program:

%_N_MAIN_MPF

AON

Clearance control ON

... G1 X100 F1000

AOFF

Clearance control OFF

M30

Notes

Position offset in the basic coordinate system

With the system variable $AA_OFF[axis] on overlaid movement of each axis in the channel is possible. It acts as a position offset in the basic coordinate system.

The position offset programmed in this way is overlaid immediately in the axis concerned, whether the axis is being moved by the program or not.

From SW 4 upwards, it is possible to limit the absolute value to be corrected (real-time variable output) to the variable in setting data SD 43350: AA_OFF_LIMIT.

The manner of overlaying the distance is defined in machine data MD 36750: AA_OFF_MODE:

0 Proportional valuation
1 Integrating valuation

With system variable $AA_OFF_LIMIT[axis] a directional scan to see whether the offset value is within the limits is possible. These system variables can be scanned from synchronized actions and, when a limit value is reached, it is possible to stop the axis or set an alarm.

0 Offset value not within limits
1 Limit of offset value reached in the positive direction
-1 Limit of the offset value reached in the negative direction
10.4.12 Online tool offset FTOC

Programming

FTOC(Polynomial_No., RV, Length1_2_3 or Radius4, channel, spindle)

Explanation

<table>
<thead>
<tr>
<th>Polynomial_No.</th>
<th>For polynomial defined with FCTDEF, see Subsection &quot;Polynomial definition&quot; in this Section.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RV</td>
<td>Real-time variable for which a function value for the specified polynomial is to be calculated.</td>
</tr>
<tr>
<td>Length1_2_3</td>
<td>Length compensation ($TC_DP1 to 3) or radius compensation to which the calculated function value is added.</td>
</tr>
<tr>
<td>Radius4</td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>Number of the channel in which the offset is active. No specification is made here for an offset in the active channel. FTOCON must be activated in the target channel.</td>
</tr>
<tr>
<td>Spindle</td>
<td>Only specified if it is not the active spindle which is to be compensated.</td>
</tr>
</tbody>
</table>

Function

FTOC permits overlaid movement for a geometry axis after a polynomial programmed with FCTDEF depending on a reference value that might, for example, be the actual value of an axis. This means that you can also program modal, Online tool compensations or clearance controls as synchronized actions.

Application

Machining of a workpiece and dressing of a grinding wheel in the same channel or in different channels (machining and dressing channel). The supplementary conditions and specifications for dressing grinding wheels apply to FTOC in the same way that they apply to tool offsets using PUTFTOCCF. For further information, please refer to Chapter 5 "Tool Offsets".
Programming example

In this example, we want to compensate for the length of the active grinding wheel.

```plaintext
% N_DRESS_MPF

FCTDEF(1,-1000,1000,-$AA_IW[V],1)  # Define function

ID=1 DO FTOC(1,$AA_IW[V],3,1)  # Select online tool compensation:

WAITM(1,1,2)  # Synchronization with machining channel

G1 V-0.05 F0.01 G91  # Infeed movement for dressing

G1 V-0.05 F0.02

...  # Deselect online offset

CANCEL(1)

...
```

---

© Siemens AG, 2002. All rights reserved
SI\,NUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition

10-429
10.4.13 Positioning movements

Function

Axes can be positioned completely unsynchronized with respect to the parts program from synchronized actions. The programming of positioning axes from synchronized actions is advisable for cyclic sequences or operations that are strongly dependent on events. Axes programmed from synchronized actions are called command axes.

In SW 5 and higher, G codes G70/G71/G700/G710 can be programmed in synchronized actions. They can be used for defining the measuring system for positioning tasks in synchronized actions.

References: /PG/ Chapter 3 "Specifying paths" /FBSY/ "Starting Command Axes"

The measuring system is defined using G70/G71/G700/G710.

By programming the G functions in the synchronized action, the INCH/METRIC evaluation for the synchronized action can be defined independently of the parts program context.

Example 1

| N100 | R1=0 |
| N110 | G0 X0 Z0 |
| N120 | WAITP(X) |
| N130 | ID=1 WHENEVER $R==1 DO POS[X]=10 |
| N140 | R1=1 |
| N150 | G71 Z10 F10 |
| N160 | Z=10mm X=254mm |
| N170 | G71 Z10 F10 |
| N180 | M30 |

The program environment affects the positioning travel of the positioning axis (no G function in the action part of the synchronized action)
Example 2

G71 in the action part of the synchronized action clearly determines the positioning travel of the positioning axis (metric), whatever the program environment.

```
N100 R1=0
N110 G0 X0 Z0
N120 WAITP(X)
N130 ID=1 WHENEVER $R==1 DO G71 POS[X]=10
N140 R1=1
N150 G71 Z10 F10 Z=10mm X=10mm
N160 G70 Z10 F10 Z=254mm X=10mm (X is always positioned to 10mm)
N170 G71 Z10 F10 Z=10mm X=10mm
N180 M30
```

Programming example

Disabling a programmed axis motion

If you do not want the axis motion to start at the beginning of the block, the override for the axis can be held at 0 until the appropriate time from a synchronized action.

```
WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0
   G01 X10 Y25 F750 POS[W]=1500
   FA=1000
   ;The positioning axis is halted as long as digital input 1 =0
```
10.4 Actions within synchronized actions

10.4.14 Position axis POS

Function

POS[axis]=value

Unlike programming from the parts program, the positioning axis movement has no effect on execution of the parts program.

Explanation

<table>
<thead>
<tr>
<th>Axis:</th>
<th>Name of the axis to be traversed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td>The value to traverse by (depending on traverse mode)</td>
</tr>
</tbody>
</table>

Programming example

ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100

Axis U is moved incrementally from the control zero by 100 (inch/mm) or to position 100 (inch/mm) independently of the traversing mode.


Axis U moved by a path calculated from real-time variables.

10.4.15 Start/stop axis MOV

Programming

MOV [Axis]=value

Explanation

<table>
<thead>
<tr>
<th>Axis:</th>
<th>Name of the axis to be started</th>
</tr>
</thead>
</table>
| Value:        | Start command for traverse/stop motion.
|               | The sign determines the direction of motion.
|               | The data type for the value is INTEGER. |
| Value >0 (usually +1): | Positive direction |
| Value <0 (usually -1): | Negative direction |
| Value ==0:     | Stop axis movement |
Function

With MOV[axis]=value it is possible to start a command axis without specifying an end position. The axis is moved in the programmed direction until another movement is set by another motion or positioning command or until the axis is stopped with a stop command.

Programming example

... DO MOV[U]=0
Axix U is stopped

Note

If an indexing axis is stopped with MOV[Axis]=0, the axis is halted at the next indexing position.

10.4.16 Axial feed FA

Programming example

FA[axis]=feedrate

ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990
;Define fixed feedrate value
;Calculate feedrate value from real-time variables
10.4.17 SW limit switch

**Function**

The working area limitation programmed with G25/G26 is taken into account for the command axes depending on the setting data SA_WORKAREA_PLUS_ENABLE. Switching the working area limitation on and off with G functions WALIMON/WALIMOF in the parts program has no effect on the command axes.

10.4.18 Axis coordination

**Function**

Typically, an axis is either moved from the parts program in the motion block or as a positioning axis from a synchronized action. If the same axis is to be traversed alternately from the parts program as a path or positioning axis and from synchronized actions, however, a coordinated transfer takes place between both axis movements. If a command axis is subsequently traversed from the parts program, preprocessing must be reorganized. This, in turn, causes an interruption in the parts program processing comparable to a preprocessing stop.
Programming example

Move the X axis from either the parts program or the synchronized actions:

```
N10 G01 X100 Y200 F1000  ; X axis programmed in the parts program
...
N20 ID=1 WHEN $A_IN[1]==1 DO
  POS[X]=150  FA[X]=200
  ; Starting positioning from the synchronized action if a digital input is set
...
CANCEL(1)  ; Deselect synchronized action
...
N100 G01 X240 Y200 F1000
  ; X becomes the path axis; before motion, delay occurs because of axis transfer
  ; if digital input was 1 and X was positioned from the synchronized action.
```

Programming example

Change traverse command for the same axis:

```
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560
  ; Start positioning from the synchronized action if a digital input >= 1
  ; Axis follows, 2nd input is set, i.e. end position and feed for axis V are continuously followed during a movement when two synchronized actions are simultaneously active.
```
10.4 Actions within synchronized actions

10.4.19 Set actual value

Function

When PRESETON (axis, value) is executed, the current axis position is not changed but a new value is assigned to it.

Notes

PRESETON can be executed from within a synchronized action in the following cases:

- Modulo rotary axes that have been started from the parts program
- All command axes that have been started from the synchronized action

Restriction:
PRESETON is not possible for axes that participate in a transformation.

Programming example

```
WHEN $AA_IM[a] >= 89.5 DO PRESETON(a4,10.5)

;Offset control zero of axis a by 10.5 length units (inch or mm) in the positive axis direction.
```

Restriction

One and the same axis can by moved from the parts program and from a synchronized action, only at different times. For this reason, delays can occur in the programming of an axis from the parts program if the same axis has been program in a synchronized action first.

If the same axis is used alternately, transfer between the two axis movements is coordinated. Parts program execution must be interrupted for that.
10.4.20 Spindle motions

Function
Spindles can be positioned completely unsynchronized with respect to the parts program from synchronized actions. This type of programming is advisable for cyclic sequences or operations that are strongly dependent on events.

Programming example
Start/stop/position spindles

| ID=1 EVERY $A_IN[1]==1 DO M3 S1000 | Set direction and speed of rotation |
| ID=2 EVERY $A_IN[2]==1 DO SPOS=270 | Position spindle |

Sequence of execution
If conflicting commands are issued for a spindle via simultaneously active synchronized actions, the most recent spindle command takes priority.

Programming example
Set direction and speed of rotation/position spindle

| ID=1 EVERY $A_IN[1]==1 DO M3 S300 | Set direction and speed of rotation |
| ID=2 EVERY $A_IN[2]==1 DO M4 S500 | Specify new direction and new speed of rotation |
| ID=3 EVERY $A_IN[3]==1 DO S1000 | Specify new speed |
| ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0 | Position spindle |
10.4.21 Coupled-axis motion TRAILON, TRAILOF

Function

Activate coupled-axis motion:

\[
\text{DO TRAILON(following axis, leading axis, coupling factor)}
\]

Deactivate coupled-axis motion:

\[
\text{DO TRAILOF(following axis, leading axis, leading axis 2)}
\]

When the coupling is activated from the synchronized action, the leading axis can be in motion. In this case the following axis is accelerated up to the set velocity. The position of the leading axis at the time of synchronization of the velocity is the starting position for coupled-axis motion. The functionality of coupled-axis motion is described in the Section "Path traversing behavior".

Programming example

Activate 1st combined axis pair when digital input is 1

\[
\text{\$A\_IN[1]==0 DO TRAILON(Y,V,1)}
\]

Activate 2nd combined axis pair

\[
\text{\$A\_IN[2]==0 DO TRAILON(Z,W,-1)}
\]

Infeed of Z and W axes in opposite axis directions

\[
\text{G0 Z10}
\]

Infeed of Y and V axes in same axis directions

\[
\text{G0 Y20}
\]

Superimpose dependent and independent movement of coupled-motion axis "V"

\[
\text{G1 Y22 V25}
\]

Deactivate 1st coupled axis

\[
\text{TRAILOF(Y,V)}
\]

Deactivate 2nd coupled axis

\[
\text{TRAILOF(Z,W)}
\]
10.4.22 Leading value coupling LEADON, LEADOF

Function

The axial leading value coupling can be programmed in synchronized actions without restriction.

Activate axial leading value coupling:

```plaintext
...DO LEADON (FA, LA, NR)
```

Where:
- FA: Following axis
- LA: Leading axis
- NR: Number of stored curve table

Deactivate axial leading value coupling:

```plaintext
...DO LEADOF (FA, LA)
```

Where:
- FA: Following axis
- LA: Leading axis

The axis to be coupled is released for synchronized action access by invoking the RELEASE function for the axis.

Example:

```plaintext
RELEASE (XKAN)
ID=1 every SR1==1 to LEADON(CACH,XKAN,1)
```

Programming example

On-the-fly parting

A continuous material that runs continuously through the work area of parting device is to be separated into pieces of equal length.

X axis: Axis in which the continuous material runs. WCS

X1 axis: Machine axis of the continuous material, MCS

Y axis: Axis in which the parting device "travels" with the continuous material

It is assumed that the positioning and control of the parting tool is controlled by the PLC. The signals of the PLC interface can be evaluated for the purpose of determining the degree of synchronism between the continuous material and the parting tool.

Actions
- Activate coupling, LEADON
- Deactivate coupling, LEADOF
- Set actual value, PRESETON
10.4 Actions within synchronized actions

%_N_SHEARS1_MPF
; $PATH=/_N_WCS_DIR/_N_DEMOFBE_WPD
N100 R3=1500 ; Length of a section to be parted
N200 R2=100000 R13=R2/300
N300 R4=100000
N400 R6=30 ; Start position Y axis
N500 R1=1 ; Start condition for conveyor axis
N600 LEADOF (Y,X) ; Delete any existing coupling
N700 CTABDEF(Y,X,1,0) ; Table definition
N800 X=30 Y=30 ; Value pair
N900 X=R13 Y=R13
N1000 X=2*R13 Y=30
N1100 CTABEND ; End of table definition
N1200 PRESETON(X1,0) ; PRESET to begin
N1300 Y=R6 G0 ; Start pos. Y axis, axis is linear
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PRESETON(X1,0)
; PRESET after length R3, new start following parting
N1500 RELEASE(Y)
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON(Y,X,1)
; Couple Y to X via table 1, for X < 10
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO EADOF(Y,X)
; > 30 before traversed parting distance, deactivate coupling
N2000 WAITP(X)
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 ; Place material axis in continuous motion
FA[X]=$R4
N2200 M30
10.4.23 Measurement

Compared with use in traverse blocks of the parts program, the measuring function can be activated and deactivated as required.

- Axial measurement without deletion of distance-to-go:
  \[
  \text{MEAWA}[\text{axis}]=\text{(mode, trigger event}_1, \ldots, _4)
  \]

- Continuous measurement without deletion of distance-to-go:
  \[
  \text{MEAC}[\text{axis}]=\text{(mode, measurement memory, trigger event}_1, \ldots, _4)
  \]

For further information on measuring: See Chapter 5, "Extended Measuring Function"

10.4.24 Set/clear wait marks: SETM, CLEARM (SW 5.2 and higher)

Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETM(MarkerNumber)</td>
<td>Set wait marker for channel</td>
</tr>
<tr>
<td>CLEARM(MarkerNumber)</td>
<td>Clear wait marker for channel</td>
</tr>
</tbody>
</table>

In synchronized actions, wait markers can be set or deleted for the purpose of coordinating channels, for example.

SETM
The SETM command can be written in the parts program and in the action part of a synchronized action. It sets the marker MarkerNumber for the channel in which the command executes.

CLEARM
The CLEARM command can be written in the parts program and in the action part of a synchronized action. It resets the flag MarkerNumber for the channel in which the command executes.
10.4.25 Error responses

Function
Incorrect responses can be programmed with synchronized actions by scanning status variables and triggering the appropriate actions.

Some possible responses to error conditions are:
- Stop axis: Override=0
- Set alarm: With SETAL it is possible to set cyclic alarms from synchronized actions.
- Set output
- All actions possible in synchronized actions

Programming example

\[
\text{ID}=67 \ \text{WHENEVER} \ (\AA_{\text{IM}}[X1]-\AA_{\text{IM}}[X2])<4.567 \ \text{DO} \ \AA_{\text{OVR}}[X2]=0
\]

;If the safety distance between axes X1 and X2 is too small, stop axis X2.

\[
\text{ID}=67 \ \text{WHENEVER} \ (\AA_{\text{IM}}[X1]-\AA_{\text{IM}}[X2])<4.567 \ \text{DO} \ \text{SETAL}(61000)
\]

;If the safety distance between axes X1 and X2 is too small, set an alarm.

10.4.26 Travel to fixed stop FXS and FOCON/FOCOF

Explanation
FXS and FOC in synchronized actions

<table>
<thead>
<tr>
<th>FXS[axis]</th>
<th>Selection only in systems with digital drives (FDD, MSD, HLA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FXST[axis]</td>
<td>Modification of clamping torque FXST</td>
</tr>
<tr>
<td>FXSW[axis]</td>
<td>Change of monitoring window FXSW</td>
</tr>
<tr>
<td>FOCON[axis]</td>
<td>Activation of the modal torque/force limitation</td>
</tr>
<tr>
<td>FOCOF[axis]</td>
<td>Deactivation of the torque/force limitation</td>
</tr>
<tr>
<td>FOCON/FOCOF</td>
<td>The axis is programmed in square brackets. The following are permitted:</td>
</tr>
<tr>
<td></td>
<td>– Geometry axis identifier</td>
</tr>
<tr>
<td></td>
<td>– Channel axis identifier</td>
</tr>
<tr>
<td></td>
<td>– Machine axis identifier</td>
</tr>
</tbody>
</table>
**Function**

The commands for **travel to fixed stop** are programmed in synchronized actions/technology cycles with the parts program commands FXS, FXST and FXSW.

Activation can take place without movement; the torque is immediately limited. As soon as the axis moves in relation to the setpoint, fixed stop is monitored.

**Travel with limited torque/force (FOC):**

This function allows torque/force to be changed at any time via synchronized actions and can be activated modally or non-modally.

**Notes**

**Multiple activation**

The function must only be activated once. If incorrect programming activates the function again although it has already been activated (FXS[axis]=1), alarm 20092 "Travel to fixed stop still active" is output.

Programming code that scans $AA_FXS[] or a separate flag (here R1) in the condition will ensure that the function is not activated more than once.

Parts program extract:

```
N10 R1=0
N20 IDS=1 WHENEVER ($R1==0 AND $AA_IW[AX3] > 7) DO R1=1 FXST[AX1]=12
```

**Block-related synchronized actions:**

Travel to fixed stop can be activated during an approach motion by programming a block-related synchronized action.

Programming example:

```
N10 G0 G90 X0 Y0
N20 WHEN $AA_IW[X] > 17 DO FXS[X]=1 ;If X reaches a position greater than 17mm
N30 G1 F200 X100 Y110 ;FXS is activated
```

**Static and block-related synchronized actions:**

The same commands FXS, FXST and FXSW can be used in static and block-related synchronized actions as in normal parts program execution. The values that are assigned can be generated by calculation.
Programming example

Travel to fixed stop (FXS)
Triggered by a synchronized action

Y axis:

N10  IDS=1  WHENEVER  (($R1==1)  AND  ($AA_FXS[Y]==0))  DO
$R1=0  FXS[Y]=1  FXST[Y]=10
FA[Y]=200  POS[Y]=150

; By setting $R1=1, FXS is activated for
; axis Y, the effective torque is reduced to
; 10% and a traverse motion is initiated
; in the direction of the fixed stop.

N11  IDS=2  WHENEVER  ($AA_FXS[Y]==4)  DO
FXST[Y]=30

; As soon as the fixed stop is detected
; ($AA_FXS[Y]==4), torque is increased
to 30%

N12  IDS=3  WHENEVER  ($AA_FXS[Y]==1)  DO
FXST[Y]=$R0

; After the fixed stop is reached, torque
; is controlled by R0

N13  IDS=4  WHENEVER  (($R3==1)  AND  ($AA_FXS[Y]==1))  DO
FXS[Y]=0
FA[Y]=1000  POS[Y]=0

; Deselection according to
; R3 and
; return

N20  FXS[Y]=0  G0  G90  X0  Y0

; Normal program run: axis Y for

N30  RELEASE(Y)

; Enable motion in synchronized action

N40  G1  F1000  X100

; Movement of another axis

N50  ......;

N60  GET(Y)

; Put axis Y back in the path group

Programming example

Activate torque/force limitation (FOC)

N10  FOCON[X]

; Modal activation of limitation

N20  X100  Y200  FXST[X]=15

; X travels with reduced torque (15%)

N30  FXST[X]=75  X20

; Change the torque to 75%, X travels with
; this limited torque

N40  FOCOF[X]

; Deactivation of the torque limitation
10.5 Technology cycles

Function

As an action in synchronized actions, you can invoke programs. These must consist only of functions that are permissible as actions in synchronized actions. Programs structured in this way are called technology cycles.

Technology cycles are stored in the control as subroutines. As far as the user is concerned, they are called up like subroutines. Parameter transfer is not possible.

It is possible to process several technology cycles or actions in parallel in one channel.

The program end is programmed with M02/M17/M30/RET. A maximum of one axis movement per block can be programmed.

Application

Technology cycles as axis programs: Each technology cycle controls only one axis. In this way, different axis motions can be started in the same interpolation cycle under event control. The parts program is now only used for the management of synchronized actions in extreme cases.
Programming example

Axis programs are started by setting digital inputs.

Main program:

ID=1 EVERY $A_IN[1]==1 DO AXIS_X
If input 1 is at 1, axis program X starts

ID=2 EVERY $A_IN[2]==1 DO AXIS_Y
If input 2 is at 1, axis program Y starts

ID=3 EVERY $A_IN[3]==1 DO $AA_OVR[Y]=0
If input 3 is at 1, the override for axis Y is at 0

ID=4 EVERY $A_IN[4]==1 DO AXIS_Z
If input 4 is at 1, axis program Z starts

M30

Technology cycle AXIS_X:

$AA_OVR[Y]=0
M100
POS[X]=100 FA[X]=300
M17

Technology cycle AXIS_Y:

POS[Y]=10 FA[Y]=200
POS[Y]=-10
M17

Technology cycle AXIS_Z:

$AA_OVR[X]=0
POS[Z]=90 FA[Z]=250
POS[Z]=-90
M17
Technology cycles are started as soon as their conditions are fulfilled. With positioning axes, several IPO cycles are required for execution. Other functions (OVR) are executed in one cycle. In the technology cycle, blocks are executed in sequence.

**Notes**
If actions are called in the same interpolation cycle that are mutually exclusive, the action is started that is called from the synchronized action with the higher ID number.

### 10.5.1 Lock, unlock, reset: LOCK, UNLOCK, RESET

#### Programming

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK (n, n, ...)</td>
<td>Lock technology cycle, the active action is interrupted</td>
</tr>
<tr>
<td>UNLOCK (n, n, ...)</td>
<td>Unlock technology cycle</td>
</tr>
<tr>
<td>RESET (n, n, ...)</td>
<td>Reset technology cycle, the active action is interrupted</td>
</tr>
</tbody>
</table>

- n: Identification number of the synchronized action

#### Function

Execution of a technology cycle can be locked, unlocked or reset from within a synchronized action or from a technology cycle.

**Lock technology cycle, LOCK**

Technology cycles can be locked using LOCK from another synchronized action or from a technology cycle.

**Example:**

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
```
Unlock technology cycle, UNLOCK
Locked technology cycles can be unlocked again from another synchronized action/technology cycle with UNLOCK. With UNLOCK, this is continued at the current position, this also applies to an interrupted positioning procedure. 

Example:
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130  
...  
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)  
...  
N250 ID=3 WHENEVER $A_IN[3]==1 DO UNLOCK(1)

Reset technology cycle, RESET
Technology cycles can be reset using RESET from another synchronized action or from a technology cycle.

Example:
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130  
...  
N200 ID=2 WHENEVER $A_IN[2]==1 DO RESET(1)

Locking on the PLC side
Modal synchronized actions can be interlocked from the PLC with the ID numbers n=1 ... 64. The associated condition is no longer evaluated and execution of the associated function is locked in the NCK. All synchronized actions can be locked indiscriminately with one signal in the PLC interface.

Notes
A programmed synchronized action is active as standard and can be protected against overwriting/locking by a machine data setting.

Application:
It should not be possible for end customers to modify synchronized actions defined by the machine manufacturer.
10.6 Cancel synchronized action: CANCEL

**Programming**

<table>
<thead>
<tr>
<th>CANCEL(n,n,...)</th>
<th>Cancel synchronized action</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>Identification number of the synchronized action</td>
</tr>
</tbody>
</table>

**Explanation**

Modal synchronized actions with the identifier ID(S)=n can only be canceled directly from the parts program with CANCEL.

**Example:**

N100 ID=2 WHENEVER $A_IN[1]==1 DO M130

...  

N200 CANCEL(2) Cancel synchronized action No. 2

**Notes**

Incomplete movements originating from a canceled synchronized action are completed as programmed.
10.7 Supplementary conditions

- **Power ON**
  With power ON no synchronized actions are active.
  However, static synchronized actions can be activated on power ON with an asynchronous subroutine (ASUB) started by the PLC.

- **Mode change**
  Synchronized actions activated with the vocabulary word IDS remain active following a changeover in operating mode.
  All other synchronized actions become inactive following operating mode changeover (e.g. axis positioning) and become active again following repositioning and a return to automatic mode.

- **Reset**
  With NC reset, all actions started by synchronized actions are stopped. Static synchronized actions remain active. They can start new actions.
  The **RESET** command can be used from the synchronized action or from a technology cycle to reset a modally active synchronized action. If a synchronized action is reset while the positioning axis movement that was activated from it is still active, the positioning axis movement is interrupted.
  Synchronized actions of the WHEN type that have already been executed are not executed again following **RESET**.
## Response following RESET

<table>
<thead>
<tr>
<th>Synchronized action / technology cycle</th>
<th>Modal / non-modal</th>
<th>Static (IDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active actions are reset, synchronized actions are canceled</td>
<td>Active action is canceled, technology cycle is reset</td>
<td></td>
</tr>
<tr>
<td>Axis / positioning spindle</td>
<td>Movement is reset</td>
<td>Movement is reset</td>
</tr>
<tr>
<td>Speed-controlled spindle</td>
<td>$\text{SMA_SPIND_ACTIVE_AFTER_RESET)==1:}$ Spindle remains active</td>
<td>$\text{SMA_SPIND_ACTIVE_AFTER_RESET)==0:}$ Spindle is stopped.</td>
</tr>
<tr>
<td>$\text{SMA_SPIND_ACTIVE_AFTER_RESET)==0:}$ Spindle is stopped.</td>
<td>$\text{SMA_SPIND_ACTIVE_AFTER_RESET)==1:}$ Spindle remains active</td>
<td></td>
</tr>
<tr>
<td>Leading value coupling</td>
<td>$\text{SMC_RESET_MODE_MASK, Bit13 == 1:}$ Leading value coupling remains active</td>
<td>$\text{SMC_RESET_MODE_MASK, Bit13 == 0:}$ Leading value coupling is disconnected</td>
</tr>
<tr>
<td>$\text{SMC_RESET_MODE_MASK, Bit13 == 0:}$ Leading value coupling is disconnected</td>
<td>$\text{SMC_RESET_MODE_MASK, Bit13 == 1:}$ Leading value coupling remains active</td>
<td></td>
</tr>
<tr>
<td>Measuring procedures</td>
<td>Measurements started from synchronized actions are canceled.</td>
<td>Measurements started from static synchronized actions are canceled.</td>
</tr>
</tbody>
</table>

### • NC Stop

**Static** synchronized actions remain active on NC stop. Movements started from static synchronized actions are not canceled.

Synchronized actions that are **local to the program** and belong to the active block remain active, movements started from them are stopped.

### • End of program

End of program and synchronized action do not influence one another.

Current synchronized actions are completed even after end of program.

Synchronized actions active in the M30 block remain active. If you do not want them to remain active, cancel the synchronized action before end of program by pressing CANCEL (see preceding section).
### Response following end of program

<table>
<thead>
<tr>
<th>Synchronized action / technology cycle</th>
<th>Modal and non-modal are reset</th>
<th>Static (IDS) remain active</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Axis / positioning spindle</strong></td>
<td>M30 is delayed until the axis/spindle is stationary.</td>
<td>Movement continues</td>
</tr>
<tr>
<td><strong>Speed-controlled spindle</strong></td>
<td>End of program: $\text{MA_SPIND_ACTIVE_AFTER_RESET}==1$: Spindle remains active $\text{MA_SPIND_ACTIVE_AFTER_RESET}==0$: Spindle is stopped</td>
<td>Spindle remains active</td>
</tr>
<tr>
<td><strong>Leading value coupling</strong></td>
<td>$\text{MC_RESET_MODE_MASK, Bit13 == 1:}$ Leading value coupling remains active $\text{MC_RESET_MODE_MASK, Bit13 == 0:}$ Leading value coupling is disconnected</td>
<td>A coupling started from a static synchronized action remains</td>
</tr>
<tr>
<td><strong>Measuring procedures</strong></td>
<td>Measurements started from synchronized actions are canceled.</td>
<td>Measurements started from static synchronized actions remain active.</td>
</tr>
</tbody>
</table>

### Block search

Synchronized actions found during a block search are collected and evaluated on NC Start; the associated actions are then started if necessary.

Static synchronized actions are active during block search.

If polynomial coefficients programmed with FCTDEF are found during a block search, they are written directly to the setting data.

### Program interruption by asynchronous subroutine

ASUB start:

Modal and static motion-synchronized actions remain active and are also active in the asynchronized subroutine.

ASUB end:

If the asynchronized subroutine is not resumed with Repos, modal and static motion-synchronized actions that were modified in the asynchronized subroutine remain active in the main program.
• Repositioning
On repositioning REPOS, the synchronized actions that were active in the interrupted block are reactivated.
Modal synchronized actions changed from the asynchronized subroutine are not active after REPOS when the rest of the block is executed.
Polynomial coefficients programmed with FCTDEF are not affected by asynchronized subroutines and REPOS. No matter where they were programmed, they can be used at any time in the asynchronized subroutine and in the main program after execution of REPOS.

• Deselection with CANCEL
If an active synchronized action is deselected with CANCEL, this does not affect the active action. Positioning movements are terminated in accordance with programming.
The CANCEL command is used to interrupt a modally or statically active synchronized action.
If a synchronized action is canceled while the positioning axis movement that was activated from it is still active, the positioning axis movement is interrupted. If this is not required, the axis movement can be decelerated before the CANCEL command with axial deletion of distance-to-go:

Example:

```
ID=17 EVERY $A_IN[3]==1 DO POS[X]=15 FA[X]=1500 ;Start positioning axis movement
...
WHEN ... DO DELDTG(X) ;End positioning axis movement
CANCEL(1)
```

10.7 Supplementary conditions
Oscillation

11.1 Asynchronous oscillation

11.2 Oscillation controlled via synchronous actions
### 11.1 Asynchronous oscillation

#### Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSP1[axis]=</td>
<td>Position of reversal point 1</td>
</tr>
<tr>
<td>OSP2[axis]=</td>
<td>Position of reversal point 2</td>
</tr>
<tr>
<td>OST1[axis]=</td>
<td>Stopping time at reversal points in seconds</td>
</tr>
<tr>
<td>OST2[axis]=</td>
<td></td>
</tr>
<tr>
<td>FA[axis]=</td>
<td>Feed for oscillating axis</td>
</tr>
<tr>
<td>OSCTRL[axis]=</td>
<td>(Set, reset options)</td>
</tr>
<tr>
<td>OSNSC[axis]=</td>
<td>Number of spark-out strokes</td>
</tr>
<tr>
<td>OSE[axis]=</td>
<td>End position</td>
</tr>
<tr>
<td>OS[axis]=</td>
<td>1 = activate oscillation; 0 = deactivate oscillation</td>
</tr>
</tbody>
</table>

#### Function

An oscillating axis travels back and forth between two reversal points 1 and 2 at a defined feedrate, until the oscillating motion is deactivated. Other axes can be interpolated as desired during the oscillating motion. A path movement or a positioning axis can be used to achieve a constant infeed, however, there is no relationship between the oscillating movement and the infeed movement.
The oscillating axis
For the oscillating axis, the following applies:
• Any axis can be used as an oscillating axis.
• Several oscillating axes can be active simultaneously (maximum: number of positioning axes).
• Linear interpolation G1 is always active for the oscillating axis – irrespective of the G command currently valid in the program.

The oscillating axis can
• act as an input axis for a dynamic transformation
• act as a guide axis for gantry and combined-motion axes
• be traversed
  – without jerk limitation (BRISK) or
  – with jerk limitation (SOFT) or
  – with acceleration curve with a knee
    (as for positioning axes).

Oscillation reversal points
The current offsets must be taken into account when oscillation positions are defined:
• Absolute specification
  OSP1[Z]=value
  Position of reversal point = sum of offsets + programmed value
• Relative specification
  OSP1[Z]=IC(value)
  Position of reversal point = reversal point 1 + programmed value

Example:
N10 OSP1[Z]=100 OSP2[Z]=110
.
.
N40 OSP1[Z]=IC(3)
Properties of asynchronous oscillation

- Asynchronous oscillation is active beyond block limits on an axis-specific basis.
- Block-oriented activation of the oscillation movement is ensured by the parts program.
- Combined interpolation of several axes and superimposing of oscillation paths are not possible.

Setting data

The setting data necessary for asynchronous oscillation can be set in the parts program.

If the setting data are described directly in the program, the change takes effect during preprocessing. A synchronized response can be achieved by means of a STOPRE.

Example:

Oscillation with online change of reversal position

```plaintext
$SA_OSCILL_REVERSE_POS1[Z]=-10
$SA_OSCILL_REVERSE_POS2[Z]=10

G0 X0 Z0
WAITP(Z)

ID=1 WHENEVER $AA_IM[Z] < $SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0

; If the actual value of the oscillation axis has exceeded the reversal point,
; the infeed axis is stopped.

OS[Z]=1 FA[X]=1000 POS[X]=40
; Switch on oscillation
OS[Z]=0
; Switch off oscillation
M30
```
Notes on individual functions

The following addresses allow asynchronous oscillation to be activated and controlled from the parts program.

The programmed values are entered in the corresponding setting data with block synchronization during the main run and remain active until changed again.

Activate, deactivate oscillation: OS
OS[axis] = 1: Activate
OS[axis] = 0: Deactivate

WAITP (axis):
- If oscillation is to be performed with a geometry axis, you must enable this axis for oscillation with WAITP.
- When oscillation has finished, this command is used to enter the oscillating axis as a positioning axis again for normal use.

Stopping times at reversal points:
OST1, OST2

<table>
<thead>
<tr>
<th>Hold time</th>
<th>Movement in exact stop area at reversal point</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Interpolation is continued without waiting for exact stop</td>
</tr>
<tr>
<td>-1</td>
<td>Wait for exact stop coarse</td>
</tr>
<tr>
<td>0</td>
<td>Wait for exact stop fine</td>
</tr>
<tr>
<td>&gt;0</td>
<td>Wait for exact stop fine and then wait for stopping time</td>
</tr>
</tbody>
</table>

The unit for the stopping time is identical to the stopping time programmed with G4.

Note

Oscillation with motion-synchronous action and stopping times “OST1/OST2”.
When the stopping times have elapsed, the internal block change takes place during oscillation (visible at the new residual paths of the axes). When block change has been completed, the deactivation function is checked. During checking, the deactivation function is defined according to the control setting for the "OSCTRL" sequence of motions.
This timing is affected by the feedrate override. Under certain circumstances, an oscillating stroke is performed before the spark out strokes are started or the end position approached. The impression created is that the deactivation response changes. However, this is not the case.

Setting feed FA
The feedrate is the defined feedrate of the positioning axis. If no feedrate is defined, the value stored in the machine data applies.

Defining the sequence of motions: OSCTRL
The control settings for the movement are set with enable and reset options.

Reset options
These options are deactivated (only if they have previously been activated as setting options).

Set options
These options are switched over. When OSE (end position) is programmed, option 4 is implicitly activated.
### Option value

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>When the oscillation is deactivated, stop at the next reversal point (default) only possible by resetting values 1 and 2</td>
</tr>
<tr>
<td>1</td>
<td>When the oscillation is deactivated, stop at reversal point 1</td>
</tr>
<tr>
<td>2</td>
<td>When the oscillation is deactivated, stop at reversal point 2</td>
</tr>
<tr>
<td>3</td>
<td>When the oscillation is deactivated, do not approach reversal point if no spark-out strokes are programmed</td>
</tr>
<tr>
<td>4</td>
<td>Approach end position after spark-out</td>
</tr>
<tr>
<td>8</td>
<td>If the oscillation movement is canceled by deletion of the distance-to-go: then execute spark-out strokes and approach end position if appropriate</td>
</tr>
<tr>
<td>16</td>
<td>If the oscillation movement is canceled by deletion of the distance-to-go: reversal position is approached as with deactivation</td>
</tr>
<tr>
<td>32</td>
<td>New feed is only active after the next reversal point</td>
</tr>
</tbody>
</table>
| 64     | FA = 0: Path overlay is active  
          FA = 0: Speed overlay is active |
| 128    | For rotary axis DC (shortest path) |
| 256    | 0 = The sparking out stroke is a dual stroke. (default) 1 = single stroke. |

Several options are appended with plus characters.

Example:

\[
\text{OSCTRL[Z]} = (1+4,16+32+64)
\]
Programming example

Oscillating axis Z is to oscillate between 10 and 100. Approach reversal point 1 with exact stop fine, reversal point 2 with exact stop coarse. Machining takes place with feedrate 250 for the oscillating axis. At the end of the machining operation, 3 spark-out strokes must be executed and end position 200 approached with the oscillating axis.

The feed for the infeed axis is 1, the end of the infeed in the X direction is at 15.

WAITP(X,Y,Z)  Starting position
G0 X100 Y100 Z100    Switch over in positioning axis operation
N40 WAITP(X,Z)

N50 OSP1[Z]=10 OSP2[Z]=100 ->  Reversal point 1, reversal point 2
   -> OSE[Z]=200 ->    End position
   -> OST1[Z]=0 OST2[Z]=-1 ->
   -> FA[Z]=250 FA[X]=1 ->
   -> OSCTRL[Z]=(4,0) ->
   -> OSNSC[Z]=3 ->
N60 OS[Z]=1

N70 WHEN $A_IN[3]==TRUE ->  Deletion of distance-to-go
   -> DO DELDTG(X)
N80 POS[X]=15  Starting position X axis
N90 POS[X]=50
N100 OS[Z]=0  Stop oscillation
M30

-> can be programmed in a single block.
11.2 Oscillation controlled via synchronous actions

Programming:
1. Define parameters for oscillation
2. Define motion-synchronous actions
3. Assign axes, define infeed

Parameters for oscillation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSP1 (oscillating axis)</td>
<td>Position of reversal point 1</td>
</tr>
<tr>
<td>OSP2 (oscillating axis)</td>
<td>Position of reversal point 2</td>
</tr>
<tr>
<td>OST1 (oscillating axis)</td>
<td>Stopping time at reversal point 1 in seconds</td>
</tr>
<tr>
<td>OST2 (oscillating axis)</td>
<td>Stopping time at reversal point 2 in seconds</td>
</tr>
<tr>
<td>FA (OscillationAxis)</td>
<td>Feed for oscillating axis</td>
</tr>
<tr>
<td>OSCTRL (OscillationAxis)</td>
<td>Set or reset options</td>
</tr>
<tr>
<td>OSNSC (oscillating axis)</td>
<td>Number of spark-out strokes</td>
</tr>
<tr>
<td>OSE (OscillationAxis)</td>
<td>End position</td>
</tr>
<tr>
<td>WAITP (OscillationAxis)</td>
<td>Enable axis for oscillation</td>
</tr>
</tbody>
</table>

Axis assignment, infeed

| OSCILL (OscillationAxis)       | Assign infeed axis or axes for oscillating axis       |
| POSP (InfeedAxis)              | Define complete and partial infeeds (see Chapter 3)   |

OSCILL = (InfeedAxis1, InfeedAxis2, InfeedAxis3)
POSP = (Endpos, Partial length, Mode)

Endpos
Length of the partial infeed at reversal point/reversal area
Mode
Division of the complete infeed into partial infeeds

Motion-synchronous actions

WHEN... ... DO
WHENEVER ... DO
Control oscillation via synchronized actions

With this mode of oscillation, an infeed motion may only be executed at the reversal points or within defined reversal areas.

Depending on requirements, the oscillation movement can be
• continued or
• stopped until the infeed has been finished executing.

Sequence

1. Define oscillation parameters
   The parameters for oscillation should be defined before the movement block containing the assignment of infeed and oscillating axes and the infeed definition (see "Asynchronized oscillation").

2. Define motion-synchronized actions
   The following synchronization conditions can be defined:
   • Suppress infeed until the oscillating axis is within a reversal area (ii1, ii2) or at a reversal point (U1, U2).
   • Stop oscillation motion during infeed at reversal point.
   • Restart oscillation movement on completion of partial infeed.
   • Define start of next partial infeed.

3. Assign oscillating and infeed axes as well as partial and complete infeed.
Assignment of oscillating and infeed axes

OSHILL

OSHILL[oscillating axis] = (infeed axis1, infeed axis2, infeed axis3)

The axis assignments and the start of the oscillation movement are defined with the OSHILL command.

Up to 3 infeed axes can be assigned to an oscillating axis.

Before oscillation starts, the synchronization conditions must be defined for the behavior of the axes.

Define infeeds: POSP

POSP[InfeedAxis] = (EndPosition, Part, Mode)

The following are declared to the control with the POSP command:
• Complete infeed (with reference to end position)
• The length of the partial infeed at the reversal point or in the reversal area
• The partial infeed response when the end position is reached (with reference to mode)

Mode = 0 The distance-to-go to the destination point for the last two partial infeeds is divided into 2 equal steps (default setting).
Mode = 1 All partial infeeds are of equal size. They are calculated from the complete infeed.
11.2 Oscillation controlled via synchronous actions

The synchronized actions

The synchronized motion actions listed below are used for general oscillation. You are given example solutions for individual tasks which you can use as modules for creating user-specific oscillation movements.

In individual cases, the synchronization conditions can be programmed differentially.

Vocabulary words

WHEN ... DO ... when ... , do ...
WHenever ... DO whenever ... , do ...

You can implement the following functions with the language resources described in detail below:

1. Infeed at reversal point
2. Infeed at reversal area.
3. Infeed at both reversal points.
4. Stop oscillation movement at reversal point.
5. Restart oscillation movement
6. Do not start partial infeed too early.

The following assumptions are made for all examples of synchronized actions presented here:

- Reversal point 1 < reversal point 2
- Z = oscillating axis
- X = infeed axis

You will find more information on synchronized motion actions in Section 11.3.
11.2 Oscillation controlled via synchronous actions

Infeed in reversal area

The infeed motion must start within a reversal area before the reversal point is reached.

These synchronized actions inhibit the infeed movement until the oscillating axis is within the reversal area.

The following instructions are used subject to the above assumptions:

Reversal area 1:

WHENEVER $AA_IM[Z]>$SA_OSCILL_REVERSE_POS1[Z]+ii1 DO $AA_OVR[X]=0

Whenever greater than then the current position of oscillating axis in the MCS is
the start of reversal area 1
set the axial override of the infeed axis to 0%.

Reversal area 2:


Whenever less than then the current position of oscillating axis in the MCS is
the start of reversal area 2
set the axial override of the infeed axis to 0%.
11.2 Oscillation controlled via synchronous actions

Infeed at reversal point

As long as the oscillating axis has not reached the reversal point, no movement takes place on the infeed axis.

The following instructions are used subject to the above assumptions:

Reversal point 1:

WHENEVER $AA\ _{IM}[Z]\ <\ >\ $SA\_OSCILL\_REVERSE\_POS1[Z] \ DO \ $AA\_OVR[X]=0 \ ->

-> $AA\_OVR[Z]=100

Whenever the current position of oscillating axis Z in the MCS is
greater or less than the position of reversal point 1
then set the axial override of infeed axis X to 0%
and set the axial override of oscillating axis Z to 100%.

Reversal point 2:

For reversal point 2:

WHENEVER $AA\ _{IM}[Z]\ <\ >\ $SA\_OSCILL\_REVERSE\_POS2[Z] \ DO \ $AA\_OVR[X]=0 \ ->

-> $AA\_OVR[Z]=100

Whenever the current position of oscillating axis Z in the MCS is
greater or less than the position of reversal point 2
then set the axial override of infeed axis X to 0%
and set the axial override of oscillating axis Z to 100%.
Stop oscillation motion at reversal point

The oscillation axis is stopped at the reversal point, the infeed motion starts at the same time. The oscillating motion is continued when the infeed movement is complete.

This synchronized action can also be used to start the infeed movement if this has been stopped by a previous synchronized action which is still active.

The following instructions are used subject to the above assumptions:

Reversal point 1:
```
```

Whenever the current position of oscillating axis in the MCS is equal to the position of reversal point 1
then set the axial override of the oscillating axis to 0%
and set the axial override of the infeed axis to 100%.

Reversal point 2:
```
```

Whenever the current position of oscillating axis in the MCS is equal to the position of reversal point 2
then set the axial override of the oscillating axis to 0%
and set the axial override of the infeed axis to 100%.
11.2 Oscillation controlled via synchronous actions

**Online evaluation of reversal point**

If there is a main run variable coded with $S$ on the right of the comparison, then the two variables are evaluated and compared with one another continuously in the IPO cycle.

Please refer to Section "Motion-synchronized actions" for more information.

**Restart oscillation movement**

This synchronized action is used to continue the oscillating movement when the partial infeed movement is complete.

The following instructions are used subject to the above assumptions:

```
WHENEVER $AA_DTEPW[X]==0 DO $AA_OVR[Z] = 100
```

Whenever the distance-to-go for the partial infeed on infeed axis X in the WCS is equal to zero then set the axial override of the oscillating axis to 100%.
11.2 Oscillation controlled via synchronous actions

Next partial infeed

When infeed is complete, a premature start of the next partial infeed must be inhibited.
A channel-specific marker ($AC_MARKER[Index]$) is used for this purpose. It is enabled at the end of the partial infeed (partial distance-to-go $\equiv 0$) and deleted when the axis leaves the reversal area. A synchronized action is then used to inhibit the next infeed movement.

On the basis of the given assumptions, the following instructions apply for reversal point 1:

1. Set marker

$$\text{WHENEVER } \$AA_{DTEPW}[X] == 0 \text{ DO } \$AC_{MARKER}[1]=1$$

Whenever the distance-to-go for the partial infeed on infeed axis X in the WCS is equal to zero then set the marker with index 1 to 1.

2. Clear marker

$$\text{WHENEVER } \$AA_{IM}[Z]<\$SA_{OSCILL\_REVERSE\_POS1}[Z] \text{ DO } \$AC_{MARKER}[1]=0$$

Whenever the current position of oscillating axis Z in the MCS is greater or less than the position of reversal point 1 then set marker 1 to 0.

3. Inhibit infeed

$$\text{WHENEVER } \$AC_{MARKER}[1]==1 \text{ DO } \$AA_{OVR}[X]=0$$

Whenever marker 1 is equal to 1 then set the axial override of the infeed axis to 0%.
11.2 Oscillation controlled via synchronous actions

Programming example

No infeed is to take place at reversal point 1. At reversal point 2, the infeed is to start at a distance of \( ii2 \) before reversal point 2 and the oscillating axis is not to wait at the reversal point for the end of the partial infeed. Axis Z is the oscillating axis and axis X the infeed axis.

![Diagram showing oscillation and infeed areas](image)

Program extract

1. Define parameters for oscillation

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF INT ii2</td>
<td>Define variable for reversal area 2</td>
</tr>
<tr>
<td>OSP1[Z]=10 OSP2[Z]=60</td>
<td>Define reversal points 1 and 2</td>
</tr>
<tr>
<td>OST1[Z]=0 OST2[Z]=0</td>
<td>Reversal point 1: exact stop fine</td>
</tr>
<tr>
<td>FA[Z]=150 FA[X]=0.5</td>
<td>Reversal point 2: exact stop fine</td>
</tr>
<tr>
<td>OSCTRL[Z]=(2+8+16,1)</td>
<td>Oscillating axis Z feedrate, infeed axis X feedrate</td>
</tr>
<tr>
<td>OSNC[Z]=3</td>
<td>Deactivate oscillating motion at reversal point 2; after delete DTG spark-out and approach end position; after delete DTG approach reversal position</td>
</tr>
<tr>
<td>OSE[Z]=70</td>
<td>3 spark-out strokes</td>
</tr>
<tr>
<td>ii2=2</td>
<td>End position = 70</td>
</tr>
<tr>
<td>WAITP(Z)</td>
<td>Set reversal area</td>
</tr>
<tr>
<td></td>
<td>Enable oscillation for Z axis</td>
</tr>
</tbody>
</table>
2. Motion-synchronized actions

WHENEVER $AA_{IM}[Z]<$SA_{OSCI\_RE\_POS2}[Z]–ii2 DO ->
-> $AA_{OVR}[X]=0 $AC_{MARKER}[0]=0
Whenever the current position of oscillating axis Z in the MCS is
less than the start of reversal area 2
then set the axial override of infeed axis X to 0%
and set the marker with index 0 to value 0.

WHENEVER $AA_{IM}[Z]>$SA_{OSCI\_RE\_POS2}[Z] DO $AA_{OVR}[Z]=0
Whenever the current position of oscillating axis Z in the MCS is
greater or equal to the position of reversal point 2
then set the axial override of oscillating axis Z to 0%.

WHENEVER $AA_{DTEPW}[X]==0 DO $AC_{MARKER}[0]=1
Whenever the distance-to-go of the partial infeed is
equal to 0,
then set the marker with index 0 to value 1.

WHENEVER $AC_{MARKER}[0]==1 DO $AA_{OVR}[X]=0 $AA_{OVR}[Z]=100
Whenever the marker with index 0 is
equal to 1,
then set the axial override of infeed axis X to 0% in order to inhibit premature
infeed (oscillating axis Z has not yet left reversal area 2 but infeed axis X is
ready for a new infeed)
set the axial override of oscillating axis Z to 100% (this cancels the 2nd
synchronized action).

-> must be programmed in a separate block

3. Start oscillation

OSCILL[Z]=(X) POSP[X]=(5,1,1)
Start axes
Assign axis X as the infeed axis for
oscillating axis Z.
Axis X is to travel to end position 5 in
steps of 1.

M30
End of program
11.2 Oscillation controlled via synchronous actions

Notes
Punching and Nibbling

12.1 Activation, deactivation ................................................................. 12-476
  12.1.1 Language commands ................................................................. 12-476
  12.1.2 Use of M commands ................................................................. 12-479

12.2 Automatic path segmentation ..................................................... 12-480
  12.2.1 Path segmentation for path axes ............................................... 12-481
  12.2.2 Path segmentation for single axes ............................................ 12-482
  12.2.3 Programming examples ............................................................ 12-484
12.1 Activation, deactivation

12.1.1 Language commands

**Programming**

PDELAYON
PON G... X... Y... Z...
PONS G... X... Y... Z...
PDELAYOF
SON G... X... Y... Z...
SONS G... X... Y... Z...
SPOF
PUNCHACC(Smin, Amin, Smax, Amax)

**Explanation of the parameters**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PON</td>
<td>Punching on</td>
</tr>
<tr>
<td>PONS</td>
<td>Punching with leader on</td>
</tr>
<tr>
<td>SON</td>
<td>Nibbling on</td>
</tr>
<tr>
<td>SONS</td>
<td>Nibbling with leader on</td>
</tr>
<tr>
<td>SPOF</td>
<td>Punching, nibbling Off</td>
</tr>
<tr>
<td>PDELAYON</td>
<td>Punching with delay On</td>
</tr>
<tr>
<td>PDELAYOF</td>
<td>Punching with delay Off</td>
</tr>
<tr>
<td>PUNCHACC</td>
<td>Travel dependent acceleration PUNCHACC (Smin, Amin, Smax, Amax)</td>
</tr>
<tr>
<td>(S_{\text{min}})</td>
<td>Minimum hole spacing</td>
</tr>
<tr>
<td>(S_{\text{max}})</td>
<td>Maximum hole spacing</td>
</tr>
<tr>
<td>(A_{\text{min}})</td>
<td>The initial acceleration (A_{\text{min}}) can be greater than (A_{\text{max}})</td>
</tr>
<tr>
<td>(A_{\text{max}})</td>
<td>The end acceleration (A_{\text{min}}) can be less than (A_{\text{max}})</td>
</tr>
</tbody>
</table>

**Function**

**Punching and Nibbling, activate/deactivate, PON/SON**

The punching and nibbling functions are activated with PON and SON respectively. SPOF terminates all functions specific to punching and nibbling operations.

Modal commands PON and SON are mutually exclusive, i.e. PON deactivates SON and vice versa.
Punching and nibbling with leader, PONS/SONS
The SONS and PONS commands also activate the punching or nibbling functions. In contrast to SON/PON - stroke control on interpolation level - PONS and SONS control stroke initiation on the basis of signals on servo level. This means that you can work with higher stroke frequencies and thus with an increased punching capacity.

While signals are evaluated in the leader, all functions that cause the nibbling or punching axes to change position are inhibited. Example: Handwheel mode, changes to frames via PLC, measuring functions.

Otherwise PONS and SONS work in exactly the same way as PON and SON.

Punching with delay
PDELAYON effects a delay in the output of the punching stroke. The command is modal and has a preparatory function. It is thus generally programmed before PON. Punching continues normally after PDELAYOF.

Travel-dependent acceleration PUNCHACC
The NC command PUNCHACC(S\text{min}, A_{\text{min}}, S_{\text{max}}, A_{\text{max}}) specifies an acceleration characteristic that defines different accelerations (A), depending on the hole spacing (S). Example for PUNCHACC(2, 50, 10, 100)

Hole spacing less than 2mm:
Traversal acceleration is 50% of maximum acceleration.

Hole spacing from 2mm to 10mm:
Acceleration is increased to 100%, proportional to the spacing.

Hole spacing greater than 10mm:
Traverse at an acceleration of 100%.
### Initiation of stroke

#### Initiation of the first stroke

The instant at which the first stroke is initiated after activation of the function differs depending on whether nibbling or punching is selected:

**PON/PONS:**
- All strokes – even the one in the first block after activation – are executed at the block end.

**SON/SONS:**
- The first stroke after activation of the nibbling function is executed at the start of the block.
- Each of the following strokes is initiated at the block end.

#### Punching and nibbling on the spot

A stroke is initiated only if the block contains traversing information for the punching or nibbling axes (axes in active plane).

However, if you wish to initiate a stroke at the same position, you can program one of the punching/nibbling axes with a traversing path of 0.

### Additional notes

#### Machining with rotatable tools

Use the tangential control function if you wish to position rotatable tools at a tangent to the programmed path.
12.1.2 Use of M commands

By using macro technology, you can also use M commands instead of language commands:

<table>
<thead>
<tr>
<th>M Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M22</td>
<td>Nibbling on</td>
</tr>
<tr>
<td>M122</td>
<td>Nibbling with leader on</td>
</tr>
<tr>
<td>M25</td>
<td>Punching on</td>
</tr>
<tr>
<td>M125</td>
<td>Punching with leader on</td>
</tr>
<tr>
<td>M26</td>
<td>Punching on with delay</td>
</tr>
<tr>
<td>M20</td>
<td>Punching, nibbling off</td>
</tr>
<tr>
<td>M23</td>
<td>Punching, nibbling off</td>
</tr>
</tbody>
</table>
12.2 Automatic path segmentation

**Programming**

SPP =
SPN =

**Explanation**

<table>
<thead>
<tr>
<th>SPP</th>
<th>Size of path section (maximum distance between strokes); modal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN</td>
<td>Number of path sections per block; non-modal</td>
</tr>
</tbody>
</table>

**Function**

**Path segmentation**

When punching or nibbling is active, SPP and SPN cause the total traversing distance programmed for the path axes to be divided into a number of path sections of equal length (equidistant path segmentation). Each path segment corresponds internally to a block.

**Number of strokes**

When punching is active, the first stroke is executed at the end of the first path segment. In contrast, the first nibbling stroke is executed at the start of the first path segment.

The number of punching/nibbling strokes over the total traversing path is thus as follows:

Punching:

Number of strokes = number of path segments

Nibbling:

Number of strokes = number of path segments + 1

**Auxiliary functions**

Auxiliary functions are executed in the first of the generated blocks.
12.2.1 Path segmentation for path axes

**Sequence**

**Length of SPP path segment**

With the SPP command, you specify the maximum distance between strokes and thus the maximum length of the path segments into which the total traversing distance is to be divided.

The command is deactivated with SPOF or SPP=0.

Example:

```
N10 G1 SON X0 Y0
N20 SPP=2 X10
```

In this example, the total traversing distance of 10mm is divided into 5 path segments of 2mm (SPP=2) each.

The path segments effected by SPP are always equidistant, i.e. all segments are equal in length. In other words, the programmed path segment size (SPP setting) is valid only if the quotient of the total traversing distance and the SPP value is an integer. If this is not the case, the size of the path segment is reduced internally such as to produce an integer quotient.

Example:

```
N10 G1 G91 SON X10 Y10
N20 SPP=3.5 X15 Y15
```

When the total traversing distance is 15mm and the path segment length 3.5mm, the quotient is not an integer value (4.28). In this case, the SPP value is reduced down to the next possible integer quotient. The result in this example would be a path segment length of 3mm.
Number of SPN path segments
SPN defines the number of path segments to be generated from the total traversing distance. The length of the segments is calculated automatically.

Since SPN is non-modal, punching or nibbling must be activated beforehand with PON or SON respectively.

SPP and SPN in the same block
If you program both the path segment length (SPP) and the number of path segments (SPN) in the same block, then SPN applies to this block and SPP to all the following blocks.

If SPP was activated before SPN, then it takes effect again after the block with SPN.

Additional notes
Provided that punching/nibbling functions are available in the control, then it is possible to program the automatic path segmentation function with SPN or SPP even when the punching/nibbling functions are not in use.

12.2.2 Path segmentation for single axes
If single axes are defined as punching/nibbling axes in addition to path axes, then the automatic path segmentation function can be activated for them.

Response of single axis to SPP
The programmed path segment length (SPP) basically refers to the path axes. For this reason, the SPP value is ignored in blocks which contain a single axis motion and an SPP value, but not a programmed path axis.
If both a single axis and a path axis are programmed in the block, then the single axis responds according to the setting of the appropriate machine data.

1. Default setting
   The path traversed by the single axis is distributed evenly among the intermediate blocks generated by SPP.

   Example:
   N10 G1 SON X10 A0
   N20 SPP=3 X25 A100

   As a result of the programmed distance between strokes of 3mm, five blocks are generated for the total traversing distance of the X axis (path axis) of 15mm. The A axis thus rotates through 20° in every block.

2. Single axis without path segmentation
   The single axis traverses the total distance in the first of the generated blocks.

3. With/without path segmentation
   The response of the single axis depends on the interpolation of the path axes:
   - Circular interpolation:  With path segmentation
   - Linear interpolation: Without path segmentation

Response to SPN
   The programmed number of path segments is applicable even if a path axis is not programmed in the same block.
   Precondition: The single axis is defined as a punching/nibbling axis.
12.2 Automatic path segmentation

12.2.3 Programming examples

Programming example 1

The programmed nibbling paths must be divided automatically into equidistant path segments.

Program extract

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100</td>
<td>G90 X130 Y75 F60 SPOF</td>
<td>Position at starting point 1</td>
</tr>
<tr>
<td>N110</td>
<td>G91 Y125 SPP=4 SON</td>
<td>Nibbling on, maximum path segment length for automatic path segmentation: 4mm</td>
</tr>
<tr>
<td>N120</td>
<td>G90 Y250 SPOF</td>
<td>Nibbling off, position at starting point 2</td>
</tr>
<tr>
<td>N130</td>
<td>X365 SON</td>
<td>Nibbling on, maximum path segment length for automatic path segmentation: 4mm</td>
</tr>
<tr>
<td>N140</td>
<td>X525 SPOF</td>
<td>Nibbling off, position at starting point 3</td>
</tr>
<tr>
<td>N150</td>
<td>X210 Y75 SPP=3 SON</td>
<td>Nibbling on, maximum path segment length for automatic path segmentation: 3mm</td>
</tr>
<tr>
<td>N170</td>
<td>G02 X-62.5 Y62.5 I J62.5 SPP=3 SON</td>
<td>Nibbling on, maximum path segment length for automatic path segmentation: 3mm</td>
</tr>
<tr>
<td>N180</td>
<td>G00 G90 Y300 SPOF</td>
<td>Nibbling off</td>
</tr>
</tbody>
</table>
Programming example 2

Automatic path segmentation is to be used to create the individual rows of holes. The maximum path segment length (SPP value) is specified in each case for segmentation purposes.

Program extract

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N100</td>
<td>G90 X75 Y75 F60 PON</td>
<td>Position at starting point 1; punching on; punch one hole</td>
</tr>
<tr>
<td>N110</td>
<td>G91 Y125 SPP=25</td>
<td>Maximum path segmentation length for automatic segmentation: 25mm</td>
</tr>
<tr>
<td>N120</td>
<td>G90 X150 SPOF</td>
<td>Punching off, position at starting point 2</td>
</tr>
<tr>
<td>N130</td>
<td>X375 SPP=45 PON</td>
<td>Punching on, maximum path segment length for automatic path segmentation: 45mm</td>
</tr>
<tr>
<td>N140</td>
<td>X275 Y160 SPOF</td>
<td>Punching off, position at starting point 3</td>
</tr>
<tr>
<td>N150</td>
<td>X150 Y75 SPP=40 PON</td>
<td>Punching on, the calculated path segment length of 37.79mm is used instead of the 40mm programmed as the path segment length.</td>
</tr>
<tr>
<td>N160</td>
<td>G00 Y300 SPOF</td>
<td>Punching off, position</td>
</tr>
</tbody>
</table>
13.1 Axis functions AXNAME, SPI, ISAXIS, AXSTRING (SW 6 and higher) ............... 13-489
13.2 Function call ISVAR ( ) (SW 6.3 and higher) ...................................................... 13-491
13.3 Learn compensation characteristics: QECLRNON, QECLRNOF ............................ 13-493
13.4 Synchronized spindle ............................................................................................ 13-495
13.5 EG: Electronic gear (SW 5 and higher) ................................................................ 13-501
13.5.1 Define electronic gear: EGDEF ......................................................................... 13-505
13.5.2 Activate electronic gear ...................................................................................... 13-506
13.5.3 Deactivate electronic gear .................................................................................. 13-510
13.5.4 Delete definition of an electronic gear ............................................................... 13-511
13.5.5 Revolutional feedrate (G95)/electronic gear (SW 5.2) ....................................... 13-511
13.5.6 Response of EG at Power ON, RESET, mode change, block search .............. 13-512
13.5.7 The electronic gear’s system variables ............................................................... 13-512
13.6 Extended stopping and retract (SW 5 and higher) ................................................ 13-513
13.6.1 Drive-independent reactions ............................................................................. 13-514
13.6.2 NC-controlled reactions ..................................................................................... 13-515
13.6.3 Possible trigger sources ..................................................................................... 13-518
13.6.4 Logic gating functions: Source/reaction operation ............................................ 13-518
13.6.5 Activation .......................................................................................................... 13-519
13.6.6 Generator operation/DC link backup ............................................................... 13-519
13.6.7 Drive-independent stop ..................................................................................... 13-520
13.6.8 Drive-independent retract ............................................................................... 13-521
13.6.9 Example: Using the drive-independent reaction ............................................... 13-521
13.7 Link communication (SW 5.2 and higher) ............................................................. 13-522
13.8 Axis container (SW 5.2 and higher) ....................................................................... 13-526
13.9 Program execution time/Workpiece counter (SW 5.2 and higher) ........................ 13-528
13.9.1 Program runtime ............................................................................................... 13-528
13.9.2 Workpiece counter ........................................................................................... 13-530
13.10 Interactive window call from parts program, command MMC (SW 4.4 and higher) ................................................................. 13-532
13.11 Influencing the motion control ............................................................................ 13-534
13.11.1 Percentage jerk correction: JERKLIM .............................................................. 13-534
13.11.2 Percentage velocity correction: VELOLIM ...................................................... 13-535
13.12 Master/slave grouping ......................................................................................... 13-536
13.1 Axis functions AXNAME, SPI, ISAXIS, AXSTRING (SW 6 and higher)

Programming

AXNAME("TRANSVERSE AXIS")
AX[AXNAME("string")]
AXSTRING ( (SPI(n) )
SPI(n)(spindle number)
ISAXIS(geometry axis number)

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXNAME</td>
<td>Converts an input string to an axis identifier. The input string must contain valid axis names.</td>
</tr>
<tr>
<td>SPI</td>
<td>Converts a spindle number to an axis identifier. The parameter transferred must contain a valid spindle number.</td>
</tr>
<tr>
<td>n</td>
<td>Spindle number</td>
</tr>
<tr>
<td>AXSTRING</td>
<td>Up until SW 5, the axis index of the axis which was assigned to the spindle was output as spindle number. From SW 6 the string is output with the associated spindle number.</td>
</tr>
<tr>
<td>AX</td>
<td>Variable axis identifier</td>
</tr>
<tr>
<td>ISAXIS</td>
<td>Checks whether the specified geometry axis exists.</td>
</tr>
</tbody>
</table>

Function

AXNAME is used, for example, to create generally applicable cycles when the name of the axes are not known (see also Section 13.10. "String functions").
SPI is used, for example, when axis functions are used for a spindle, e.g. the synchronized spindle.
ISAXIS is used in universal cycles in order to ensure that a specific geometry axis exists and thus that any following $P\_AXNX call is not aborted with an error message.

(SW 6 and higher)

Extensions SPI(n):
The axis function SPI(n) can now also be used for reading and writing frame components, for example, for writing frames with syntax

$S\_PFRAME[SPI](1),TR]=2.22.$
13.1 Axis functions AXNAME, SPI, ISAXIS, AXSTRING

Additional programming of the axis position via address AX[SPI(1)] = <axis position> allows an axis to be traversed.

**Troubleshooting** for AXSTRING( SPI(n) )

When programming AXSTRING( SPI(n) ) up to SW 5 the axis index of the axis which was assigned to the spindle was output as spindle number.

Example:
Spindle 1 is assigned to the 5th axis.
($MA_SPIND_ASSIGN_TO_MACHAX[AX5]=1),
AXSTRING( SPI(1) ) returns the incorrect string "S_4".

With SW 6 and higher, AXSTRING[ SPI(n) ] will output the string "S_n".

Example:
AXSTRING( SPI(2) ) returns string "S_2".

**Programming example**

Move the axis defined as a facing axis.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVRA[AXNAME(“Transverse axis”)]=10</td>
<td>Transverse axis</td>
</tr>
<tr>
<td>AX[AXNAME(“Transverse axis”)]=50.2</td>
<td>Final position for transverse axis</td>
</tr>
<tr>
<td>OVRA[SPI(1)]=70</td>
<td>Override for spindle 1</td>
</tr>
<tr>
<td>IF ISAXIS(1) == FALSE GOTOF CONTINUE</td>
<td>Does abscissa exist?</td>
</tr>
<tr>
<td>AX[SP_AXN1]=100</td>
<td>Move abscissa</td>
</tr>
</tbody>
</table>

CONTINUE:
13.2 Function call ISVAR ( ) (SW 6.3 and higher)

Programming

ISVAR ("variable identifier")
ISVAR (identifier, [value, value])

Explanation of the commands

<table>
<thead>
<tr>
<th>Variable identifiers</th>
<th>Transfer parameter of type string can be undimensioned, 1-dimensional, or 2-dimensional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Identifier with a known variable with or without an array index as machine data, setting data, system variable, or general variable</td>
</tr>
<tr>
<td>Value</td>
<td>Function value of type BOOL</td>
</tr>
</tbody>
</table>

Structure

The transfer parameter can have the following structure:

1. Undimensioned variable:
   - identifier
2. 1-dimensional variable without array index:
   - identifier[
3. 1-dimensional variable with array index:
   - identifier[value]
4. 2-dimensional variable without array index:
   - identifier[value]
5. 2-dimensional variable with array index:
   - identifier[value, value]

Function

The ISVAR command is a function as defined in the NC language with a:

- Function value of type BOOL
- Transfer parameter of type STRING

The ISVAR command returns TRUE, if the transfer parameter contains a variable known in the NC (machine data, setting data, system variable, general variables such as GUD's).
13.2 Function call ISVAR () (SW 6.3 and higher)

Checks
The following checks are made in accordance with the transfer parameter:
- Does the identifier exist
- Is it a 1- or 2-dimensional array
- Is an array index permitted

Only if all these checks have a positive result will TRUE be returned. If a check has a negative result or if a syntax error has occurred, it will return FALSE.

Axial variables are accepted as an index for the axis names but not checked.

Examples:
DEF INT VAR1
DEF BOOL IS_VAR=FALSE
N10 IS_VAR=ISVAR("VAR1")
DEF REAL VARARRAY[10,10]
DEF BOOL IS_VAR=FALSE
N20 IS_VAR=ISVAR("VARARRAY[,]")
N30 IS_VAR=ISVAR("VARARRAY")
N40 IS_VAR=ISVAR("VARARRAY[8,11]")
N50 IS_VAR=ISVAR("VARARRAY[8,8]")
N60 IS_VAR=ISVAR("VARARRAY[,8]")
N70 IS_VAR=ISVAR("VARARRAY[8,]")

DEF BOOL IS_VAR=FALSE
N100 IS_VAR=ISVAR("$MC_GCODE_RESET_VALUES[1]")

DEF BOOL IS_VAR=FALSE
N10 IS_VAR=ISVAR("$P_EP")
N10 IS_VAR=ISVAR("$P_EP[X]")
13.3 Learn compensation characteristics: QECLRNON, QECLRNOF

Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QECLRNON</td>
<td>Activate &quot;Learn quadrant error compensation&quot; function</td>
</tr>
<tr>
<td>(axis.1,...4)</td>
<td></td>
</tr>
<tr>
<td>QECLRNOF</td>
<td>Deactivate &quot;Learn quadrant error compensation&quot; function</td>
</tr>
</tbody>
</table>

Function

Quadrant error compensation (QEC) reduces contour errors that occur on reversal of the traversing direction due to mechanical non-linearities (e.g. friction, backlash) or torsion.

On the basis of a neural network, the optimum compensation data can be adapted by the control during a learning phase in order to determine the compensation characteristics automatically.

Learning can take place simultaneously for up to four axes.

Sequence

The traversing movements of the axes required for the learning process are generated with the aid of an NC program. The learning movements are stored in the program in the form of a learning cycle.

First teach-in

Sample NC programs contained on the disk of the standard PLC program are used to teach the movements and assign the QEC system variables in the initial learning phase during startup of the control.
Subsequent learning

The learnt characteristics can be optimized with subsequent learning. The data stored in the user memory are used as the basis for optimization.

Optimization is performed by adapting the sample NC programs to your needs. The parameters of the learning cycle (e.g. QECLRN.SPF) can also be changed for optimization:

- Set "Learn mode" = 1
- Reduce "Number of learn passes" if required
- Activate "Modular learning" if required and define area limits.

Activate learning process: QECLRNON

The actual learning process is activated in the NC program with the command QECLRNON and specification of the axes:

QECLRNON (X1, Y1, Z1, Q)

Only if this command is active are the quadrants changed.

Deactivate learning process: QECLRNOF

When the learning movements for the desired axes are complete, the learning process is deactivated simultaneously for all axes with QECLRNOF.
13.4 Synchronized spindle

Programming

COUPDEF (FS,LS,SF,SL, block change beh., coupling)
COUPDEL (FS,LS)
COUPRES (FS,LS)
COUPOF (FS,LS,PS,POS)
WAITC (FS, block ratio, LS, block ratio.)

Explanation of the commands

COUPDEF Define/change user coupling
COUPON Activate coupling
COUPOF Deactivate coupling
COUPRES Reset coupling parameters
COUPDEL Delete user-defined coupling
WAITC Wait for synchronism condition

Explanation of the parameters

FS, LS Name of following and leading spindle; specified with spindle number: e.g. S2
SF, SL Speed ratio parameter for following spindle and leading spindle Default setting = 1.0; specification of denominator optional
Block change behavior:
- "NOC" immediate (default)
- "FINE" in response to "Synchronization run fine"
- "COARSE" in response to "Synchronization run coarse"
- "IPOSTOP" in response to IPOSTOP (i.e. after setpoint synchronization run)
Coupling Coupling type: Coupling between FS and LS
- "DV" Setpoint coupling (default)
- "AV" Actual-value coupling
FS Angle offset between leading and following spindles
POSFS, POSLS Deactivation positions of following and leading spindles
### Function

In synchronized mode, there is a leading spindle (LS) and a following spindle (FS). They are referred to as the **synchronous spindle pair**. The following spindle follows the movements of the leading spindle when the coupling is active (synchronized mode) in accordance with the functional relationship specified in the parameters.

This function enables turning machines to perform workpiece transfer from spindle 1 to spindle 2 on-the-fly, e.g. for final machining. This avoids downtime caused, for example, by rechucking.

The transfer of the workpiece can be performed with:
- Speed synchronism \( n_{FS} = n_{LS} \)
- Position synchronism \( \phi_{FS} = \phi_{LS} \)
- Position synchronism with angular offset \( \phi_{FS} = \phi_{LS} + \Delta \phi \)

A speed ratio \( k_u \) can also be specified between the main spindle and a "tool spindle" for multi-edge machining (polygon turning).

The synchronized spindle pair can be defined permanently for each machine with channel-specific machine data or defined by the user in the CNC parts program.

Up to two synchronized spindle pairs can be operated simultaneously on each NC channel.
13.4 Synchronized spindle

Sequence

Define synchronized spindle pair Options

- Fixed definition of coupling:
  - The leading and following spindle are defined in machine data.
  - With this coupling, the machine axes defined for the LS and FS cannot be changed from the NC parts program. The coupling can nevertheless be parameterized in the NC parts program by means of COUPDEF (on condition that no write protection is valid).

- User-defined coupling:
  - The language instruction COUPDEF can be used to create new couplings and change existing ones in the NC parts programs. If a new coupling relationship is to be defined, any existing user-defined coupling must be deleted with COUPDEL.

Define new coupling COUPDEF

- The following paragraphs define the parameters for the predefined subroutine:
  - COUPDEF (FS,LS,SRFS,SRLS, block change beh., coupling)

Following and leading spindles: FS and LS

- The axis names FS and LS are used to identify the coupling uniquely.
- They must be programmed for each COUP statement. Further coupling parameters only need to be defined if they are to be changed (modal scope).

Example:

```
N... COUPDEF(S2,S1,Ü FS,Ü LS)
```

Meaning:

- S2 = following spindle, S1 = leading spindle
Positioning the following spindle: Options
When the synchronized spindle coupling is active, following spindles can also be positioned within the ±180° range independently of the motion initiated by the master spindle.

Positioning SPOS
The following spindle can be interpolated with SPOS=…

Please refer to Programming Guide "Fundamentals" for more information about SPOS.

Example:
N30  SPOS[2]=IC(-90)

FA, ACC, OVRA:
Speed, acceleration
The position speeds and acceleration rates for following spindles can be programmed with FA[SPI(Sn)] or FA[Sn], ACC[SPI(Sn)] or ACC[Sn] and OVRA[SPI(n)] or OVRA[Sn] (see Programming Guide, Fundamentals). "n" stands for spindle number 1...n.

Programmable block change WAITC
WAITC can be used to define the block change behavior with various synchronism conditions (coarse, fine, IPOSTOP) for continuation of the program, e.g. after changes to coupling parameters or positioning operations.

WAITC causes a delay in the insertion of new blocks until the appropriate synchronism condition is fulfilled, thereby allowing the synchronized state to be processed faster.

If no synchronism conditions are specified, then the block change behavior programmed/configured for the relevant coupling applies.
Examples:
	n200 WAITC
Wait for synchronism conditions for all active slave spindles without specification of these conditions.

\[ n300 \text{ WAITC}(S2, "FINE", S4, "COARSE") \]
Wait for the specified "Coarse" synchronism conditions for slave spindles S2 and S4.

\begin{itemize}
  \item \textbf{Speed ratio} \( k_U \)
  
  The speed ratio is defined with parameters for FS (numerator) and LS (denominator).

  Options:
  \begin{itemize}
    \item The following and leading spindles rotate at the same speed \( n_{FS} = n_{LS} ; \text{SR}_T \text{ positive} \)
    \item Rotation in the same or opposite direction \( \text{SR}_T \text{ negative} \) between LS and FS
    \item The following and leading spindles rotate at different speeds
      \[ n_{FS} = k_U \cdot n_{LS} ; k_U \neq 1 \]
      Application: Multi-sided turning
  \end{itemize}

\end{itemize}

Example:
\[ n_{\ldots} \text{ COUPDEF}(S2, S1, 1.0, 4.0) \]
Meaning:
Following spindle S2 and leading spindle S1 rotate at a speed ratio of 0.25.

\begin{itemize}
  \item The numerator must be programmed. If no numerator is programmed, "1" is taken as the default.
  \item The speed ratio can also be changed on-the-fly, when the coupling is active.
\end{itemize}
Block change behavior

The following options can be selected during definition of the coupling to determine when the block change takes place:

- **"NOc"** Immediately (default)
- **"FNe"** At "Synchronization fine"
- **"COARSe"** At "Synchronization coarse"
- **"IPOSTOP"** At IPOSTOP (i.e. after synchronization on the setpoint side)

It is sufficient to specify the characters typed in bold when specifying the block change method.

The block change method is modal!

Coupling type

- **"DV"** Setpoint coupling between FS and LS (default)
- **"AV"** Actual-value coupling between FS and LS

The coupling type is modal.

**Notice**

The coupling type may be changed only when the coupling is deactivated!
Activate synchronized mode

- Fastest possible activation of coupling with any angle reference between LS and FS:

\[ N \ldots \text{COUPON (S2, S1)} \]

- Activation with angular offset \( \text{POS}_{FS} \)
  Position-synchronized coupling for profiled workpieces.
  \( \text{POS}_{FS} \) refers to the 0° position of the lead spindle in the positive direction of rotation.

Value range \( \text{POS}_{FS} : 0° \ldots 359.999° \):

\[ \text{COUPON (S2,S1,30)} \]

You can use this method to change the angle offset even when the coupling is already active.

Deactivate synchronized mode \( \text{COUPOF} \)
Three variants are possible:

- For the fast possible activation of the coupling and immediate enabling of the block change:

\[ \text{COUPOF (S2,S1)} \]

- After the deactivation positions have been crossed; the block change is not enabled until the deactivation positions \( \text{POS}_{FS} \) and, where appropriate, \( \text{POS}_{LS} \) have been crossed.

Value range \( 0° \ldots 359.999° \):

\[ \text{COUPOF (S2,S1,150)} \]
\[ \text{COUPOF (S2,S1,150,30)} \]
Delete couplings, COUPDEL
An existing user-defined synchronized spindle coupling must be deleted if a new coupling relationship is to be defined and all user-configurable couplings (1 or 2) are already defined.

\[ N \ldots \text{COUPON} (S2, S1) \]

SPI(2) = following spindle, SPI(1) = leading spindle

A coupling can only be deleted if it has been deactivated first (COUPOF).

A permanently configured coupling cannot be deleted by means of COUPDEL.

Reset coupling parameters, COUPRES
Language instruction "COUPRES" is used to
- activate the parameters stored in the machine data and setting data (permanently defined coupling) and
- activate the presettings (user-defined coupling)

The parameters programmed with COUPDEF (including the transformation ratio) are subsequently deleted.

\[ N \ldots \text{COUPRES} (S2, S1) \]

S2 = following spindle, S1 = leading spindle
System variables

Current coupling status following spindle
The current coupling status of the following spindle can be read in the NC parts program with the following axial system variable:

$AA_COUP_ACT[FS]

FS = axis name of the following spindle with spindle number, e.g. S2.

The value which is read has the following meaning for the following spindle:
0: No coupling active
4: synchronized spindle coupling active

Current angular offset
The setpoint of the current position offset of the FS to the LS can be read in the parts program with the following axial system variable:

$AA_COUP_OFFS[S2]

The actual value for the current position offset can be read with:

$VA_COUP_OFFS[S2]

FS = axis name of the following spindle with spindle number, e.g. S2.

When the controller has been disabled and subsequently re-enabled during active coupling and follow-up mode, the position offset when the controller is re-enabled is different to the original programmed value. In this case, the new position offset can be read and, if necessary, corrected in the NC parts program.
13.4 Synchronized spindle

**Programming example**

Working with master and slave spindles.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N05 M3 S3000 M2=4 S2=500</td>
<td>Master spindle rotates at 3000rpm, slave spindle at 500rpm</td>
</tr>
<tr>
<td>N10 COUPDEF (S2, S1, 1, 1, &quot;NOC&quot;, &quot;Dv&quot;)</td>
<td>Def. of coupling, can also be configured</td>
</tr>
<tr>
<td>N70 SPCON</td>
<td>Include master spindle in position control (setpoint coup.)</td>
</tr>
<tr>
<td>N75 SPCON(2)</td>
<td>Include slave spindle in position control</td>
</tr>
<tr>
<td>N80 COUPOF (S2, S1, 45)</td>
<td>On-the-fly coupling to offset position = 45 degrees</td>
</tr>
<tr>
<td>N200 FA [S2] = 100</td>
<td>Positioning speed = 100 degrees/min</td>
</tr>
<tr>
<td>N205 SPOS[2] = IC(-90)</td>
<td>Traverse with 90° overlay in negative direction</td>
</tr>
<tr>
<td>N210 WAITC(S2, &quot;Fine&quot;)</td>
<td>Wait for &quot;fine&quot; synchronism</td>
</tr>
<tr>
<td>N212 G1 X... Y... F...</td>
<td>Machining</td>
</tr>
<tr>
<td>N215 SPOS[2] = IC(180)</td>
<td>Traverse with 180° overlay in positive direction</td>
</tr>
<tr>
<td>N220 G4 S50</td>
<td>Dwell time = 50 revolutions of master spindle</td>
</tr>
<tr>
<td>N225 FA [S2] = 0</td>
<td>Activate configured speed (MD)</td>
</tr>
<tr>
<td>N230 SPOS[2]=IC(-7200)</td>
<td>20 rev. with configured speed in negative direction</td>
</tr>
<tr>
<td>N350 COUPOF (S2, S1)</td>
<td>Decouple on-the-fly, S=S2=3000</td>
</tr>
<tr>
<td>N355 SPOSA[2] = 0</td>
<td>Stop slave spindle at zero degrees</td>
</tr>
<tr>
<td>N360 G0 X0 Y0</td>
<td>Wait for spindle 2</td>
</tr>
<tr>
<td>N365 WAITS(2)</td>
<td>Stop slave spindle</td>
</tr>
</tbody>
</table>
13.5  EG: Electronic gear (SW 5 and higher)

Introduction

The "Electronic gear" function allows you to control the movement of a following axis according to linear traversing block as a function of up to five leading axes. The relationship between the leading axis and the following axis are defined by the coupling factor for each leading axis. The following axis motion part is calculated by an addition of the individual leading axis motion parts multiplied by their respective coupling factors. When activating an EG axis grouping, the following axis can be synchronized according to a defined position.

A gear group can be
- defined,
- activated,
- deactivated, and
- deleted
from the parts program.

The following axis movement can be optionally derived from
- Setpoints of the leading axes, as well as
- Actual values of the leading axes.

As an expansion, with **SW 6 and higher** nonlinear relations between the leading axes and the following axis can also be achieved via curve tables (see Chapter 9). Electronic gears can be cascaded, i.e. the following axis of an electronic gear can be the leading axis for another electronic gear.

13.5.1 Define electronic gear: EGDEF

Function

An EG axis grouping is defined by specifying the following axis and a minimum of one and a maximum of five leading axes with the respective coupling type:

EGDEF (following axis, leading axis 1, coupling type 1, leading axis 2, coupling type 2, ...)

© Siemens AG, 2002. All rights reserved
13.5 EG: Electronic gear (SW 5 and higher)

**Explanation**

<table>
<thead>
<tr>
<th>Following axis</th>
<th>Axis that is influenced by the leading axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leading axis 1, ... leading axis 5</td>
<td>Axes that influence the following axis</td>
</tr>
<tr>
<td>Coupling type 1, ... coupling type 5</td>
<td>Following axis is influenced by:</td>
</tr>
<tr>
<td></td>
<td>0: actual value</td>
</tr>
<tr>
<td></td>
<td>1: setpoint</td>
</tr>
<tr>
<td></td>
<td>of the respective leading axis</td>
</tr>
</tbody>
</table>

**Programming**

```
EGDEF(C, B, 1, Z, 1, Y, 1)  
B, Z, Y influence C via setpoint
```

The coupling type does not need to be identical for all leading axes and is therefore specified for each leading axis individually.

The coupling factors are preset with zero for definition of the EG coupling group.

Requirement for an EG axis grouping definition:
- A following axis must not yet be defined for the coupled axes (if necessary, delete any existing one with EGDEL first).

**Note**

EGDEF triggers preprocessing stop. Gear definition with EGDEF must also be used unchanged, if with systems using SW 6 and higher, one or more leading axes influence the following axis via the curve table.

### 13.5.2 Activate electronic gear

There are 3 variants for the activation command:
- **Variant 1:**
  - The EG axis grouping is activated selectively
  - without synchronization with:
  
  ```
  EGON(FA, "Block change mode", LA1, Z1, N1, LA2, Z2, N2...LA5, Z5, N5.)
  ```

© Siemens AG, 2002. All rights reserved

SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
Explanation

**FA**

<table>
<thead>
<tr>
<th>Following axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block change mode</td>
</tr>
</tbody>
</table>

The following modes can be used:
- "NOC" Immediate block change
- "FINE" Block change occurs at "Synchronization fine"
- "COARSE" Block change occurs at "Synchronization coarse"
- "IPOSTOP" Block change occurs at setpoint synchronization run

**LA1, ... LA5**

Leading axes

**Z1, ... Z5**

Counter for coupling factor i

**N1, ... N5**

Denominator for coupling factor i

Coupling factor i = Counter i / Denominator i

You may only program the leading axes that have previously been specified with EGDEF. At least one leading axis must be programmed.

The positions of the leading axes and following axis at the time of activation are saved as "synchronized positions". The "synchronized positions" can be read via system variable $AA_EG_SYN.

- **Variant 2:**

The EG axis grouping is activated selectively with synchronization with:

`EGONSYN(FA, "Block change mode", SynPosFA[, LAi, SynPosLAi, Zi, Ni])`

Explanation

**FA**

<table>
<thead>
<tr>
<th>Following axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block change mode</td>
</tr>
</tbody>
</table>

The following modes can be used:
- "NOC" Immediate block change
- "FINE" Block change occurs at "Synchronization fine"
- "COARSE" Block change occurs at "Synchronization coarse"
- "IPOSTOP" Block change occurs at setpoint synchronization run
13.5  EG: Electronic gear (SW 5 and higher)

[, LAi, SynPosLAi, Zi, Ni]  
(do not write the square brackets)
min. 1, max. 5 sequences of:

<table>
<thead>
<tr>
<th>Leading axes</th>
<th>Synchronized position for i-th leading axis</th>
<th>Counter for coupling factor i</th>
<th>Denominator for coupling factor i</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA1, ... LA5</td>
<td>SynPosLAi</td>
<td>Z1, ... Z5</td>
<td>N1, ... N5</td>
</tr>
</tbody>
</table>

**Variant 3:**
The EG axis grouping is activated selectively with synchronization. The **approach mode** is specified with:

EGONSYNE(FA, "Block change mode", SynPosFA, approach mode [, LAi, SynPosLAi, Zi, Ni])

**Explanation**

The parameters are the same as for variation 2 as regards:

<table>
<thead>
<tr>
<th>Approach mode</th>
<th>The following modes can be used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;NTGT&quot;</td>
<td>Approach next tooth gap time-optimized</td>
</tr>
<tr>
<td>&quot;NTGP&quot;</td>
<td>Approach next tooth gap path-optimized</td>
</tr>
<tr>
<td>&quot;ACN&quot;</td>
<td>Traverse rotary axis in negative direction absolute</td>
</tr>
<tr>
<td>&quot;ACP&quot;</td>
<td>Traverse rotary axis in positive direction absolute</td>
</tr>
<tr>
<td>&quot;DCT&quot;</td>
<td>Time-optimized to programmed synchronized position</td>
</tr>
<tr>
<td>&quot;DCP&quot;</td>
<td>Path-optimized to programmed synchronized position</td>
</tr>
</tbody>
</table>
Variation 3 only effects modulo following axes coupled to modulo leading axes. Time optimization takes account of velocity limits of the following axis. The tooth distance (deg.) is calculated like this: 360 * Zi/Ni. If the following axis is stopped at the time of calling, path optimization returns responds identically to time optimization. If the following axis is already in motion, NTGP will synchronize at the next tooth gap irrespective of the current velocity of the following axis.

If the following axis is already in motion, NTGT will synchronize at the next tooth gap depending on the current velocity of the following axis. The axis is also decelerated, if necessary.

**SW 6**

If a **curve table** is used for one of the leading axes, then you must set:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ni</td>
<td>the denominator for the coupling factor for linear couplings must be set to 0. (Denominator 0 is illegal for linear couplings.)</td>
</tr>
<tr>
<td>Zi</td>
<td>is to be interpreted as the number of the curve table to be used. The curve table with the specified number must already be defined when the control is switched on.</td>
</tr>
<tr>
<td>LAi</td>
<td>Specification of the leading axis corresponds to the leading axis specification with coupling via coupling factor (linear coupling).</td>
</tr>
</tbody>
</table>

For more information about using curve tables and cascading and synchronizing electronic gears, please refer to:

/ FB/ M 3, Coupled Motion and Leading Value Coupling

It is only permissible to program leading axes that have previously been specified with EGDEF.
Via the programmed "synchronized positions" for the following axis (SynPosFA) and for the leading axes (SynPosLA), positions are defined in which the coupling group is valid as synchronized. If the electronic gear is not in synchronized state when it is activated, the following axis will traverse to its defined synchronized position.

If modulo axes are contained in the coupling group, their position values are modulus-reduced. This ensures that the next possible synchronized position is approached (so-called relative synchronization: e.g. the next tooth gap). The synchronized position is only approached if "Enable following axis override" interface signal DB(30 + axis number), DBX 26 bit 4 is issued for the following axis. If it is not issued, the program stops at the EGONSYN block and self-clearing alarm 16771 is output until the above mentioned signal is set.

### 13.5.3 Deactivate electronic gear

There are three different ways to deactivate an active EG axis grouping.

**Variant 1:**

EGOFS(following axis)

The electronic gear is deactivated. The following axis is decelerated until it is motionless. The call triggers preprocessing stop.

**Variant 2:**

EGOFS(following axis, leading axis 1, ... leading axis 5)

This command parameter setting make it possible to selectively remove the control the individual leading axes have over the following axis' motion.

At least one leading axis must be specified. The influence of the specified leading axes on the following axis is selectively disabled. The call triggers preprocessing stop. If leading axes are still active, the following axis will continue to operate under their control. If all leading axis influences have been disabled in this manner, the following axis is decelerated until it reaches a standstill.
13.5 EG: Electronic gear (SW 5 and higher)

Variant 3: EGOF(Following spindle)

The electronic gear is deactivated. The following spindle continues to operate with the current speed that was valid at the time of deactivation. The call triggers preprocessing stop.

Note

This function is only allowed for spindles.

13.5.4 Delete definition of an electronic gear

An EG axis grouping must be deactivated as described in the preceding section before you can delete its definition.

EGDEL(Following axis)

The coupling definition of the axis grouping is deleted. Additional axis groupings can be defined by means of EGDEF until the maximum number of simultaneously activated axis groupings is reached. The call triggers preprocessing stop.

13.5.5 Revolute feedrate (G95)/electronic gear (SW 5.2)

In SW 5 and higher, using the FPR() command, it is also possible to define the following axis of an electronic gear as the axis determining the revolute feedrate. The following applies in this case:

- The feed is dependent on the setpoint speed of the following axis of the electronic gear.
- The setpoint speed is calculated from the speed of the leading spindles and modulo leading axes (that are not path axes) and their assigned coupling factors.
- Speed parts of linear or non-modulo leading axes and overlaid movement of the following axis are not taken into account.
13.5.6 Response of EG at Power ON, RESET, mode change, block search

After Power ON there are no active couplings.
Active couplings are retained after reset and mode change.
With block search, commands for switching, deleting and defining the electronic gear are not executed or retained, instead they are skipped.

13.5.7 The electronic gear’s system variables

By means of the electronic gear’s system variables, the parts program can determine the current states of an EG axis grouping and react to them if required.

Additional notes

The system variables for the electronic gear are listed in the Annex. They are characterized by names beginning with:

$AA_EG_ . . .
or
$VA_EG_ . . .
13.6 Extended stopping and retract (SW 5 and higher)

Function

The "Extended stopping and retract" function ESR provides a means to react flexibly to selective error sources while preventing damage to the workpiece. "Extended stopping and retract" provides the following part reactions:

- **"Extended stopping"** (independent drive, SW 5)
  is a time-delayed stop.

- **"Retract"** (independent of drive)
  means "escaping" from the machining plane to a safe retraction position. This means any risk of collision between the tool and the workpiece is avoided.

- **"Generator operation"** (independent of drive)
  For the cases in which the energy of the DC link is not sufficient for a safe retraction, generator operation is possible. As an independent drive mode, it provides the drive DC link with the necessary power to perform an orderly "stop" and "retract" in the event of a power failure or similar occurrence.

From SW 6 also:

- **Extended shut down** (NC-controlled)
  is a defined, time-delayed, contour-friendly shut down controlled by the NC.

- **Retract** (NC-controlled)
  means "escaping" from the machining level to a safe retraction position under the control of the NC. This means any risk of collision between the tool and the workpiece is avoided. With gear cutting, for example, retract will cause a retraction from tooth gaps that are currently being machined.

All reactions can be used independently from one another.

For further information, see

/FB/ M3, Axis Couplings and ESR
13.6 Extended stopping and retract (SW 5 and higher)

13.6.1 Drive-independent reactions

Function

Drive-independent reactions are defined axially; if activated, each drive processes its stop/retract request independently. There is no interpolatory coupling of axes or coupling adhering to the path at stop/retract, the reference to the axes is time-controlled.

During and after execution of drive-independent reactions, the respective drive no longer follows the NC enables or NC travel commands. Power OFF/Power ON is necessary. Alarm “26110: Drive-independent stop/retract triggered” draws attention to this.

Generator operation

Generator operation is

- Configured: via MD 37500: 10
- Enabled: system variable $AA_ESR_ENABLE
- Activated: depending on the setting of the drive machine data when the voltage in the DC link falls below the value.

Retract (drive-independent)

Drive-independent retract is

- Configured: via MD 37500: 11; time specification and return velocity are set in MD, see "Example: Using the drive-independent reaction" at the end of this chapter,
- Enabled: system variable $AA_ESR_ENABLE
- Triggered: system variable $AN_ESR_TRIGGER.

Stop (independent drive)

Drive-independent stop is

- Configured: via MD 37500: 12 as well as time specification via MD;
- Enabled ($AA_ESR_ENABLE) and
- Triggered: system variable $AN_ESR_TRIGGER.
13.6.2 NC-controlled reactions

Function

Retract

Preconditions:
- the axes selected with POLFMASK
- the axis-specific positions defined with POLF
- the time window in MD 21380: ESR_DELAY_TIME1 and MD 21381: ESR_DELAY_TIME2
- the trigger via system variable $AC_ESR_TRIGGER
- the defined ESR reaction MD 37500: ESR_REACTION = 21

If system variable $AC_ESR_TRIGGER = 1 is set, and if a retract axis is configured in this channel (i.e. MD 37500: ESR_REACTION = 21) and $AA_ESR_ENABLE=1 is set for this axis, then LIFTFAST is activated in this channel.

The retract position must have been programmed in the parts program. The enabling signals must have been set for the retraction movement and must remain set.

The retracting movement configured with LFPOS, POLF for the axis/axes selected with POLFMASK replaces the path motion set in the parts program for these axes. The extended retracting movement (i.e. LIFTFAST/LFPOS triggered via $AC_ESR_TRIGGER) cannot be interrupted and can only be terminated before completion by an emergency STOP. The maximum time allowed for the retraction consists of the sum of the times specified in MD 21380: ESR_DELAY_TIME1 and MD 21381: ESR_DELAY_TIME2.
After this time has lapsed, rapid deceleration is initiated for the retracting axis too, with subsequent correction. The frame that was active when fast retraction was activated is used.

**Important:** Frames with rotation also influence the lifting direction via POLF. The NC-controlled retraction is
- configured: via MD 37500: 21 as well as 2 time specification via MD (see above);
- enabled ($AA_ESR_ENABLE) and triggered: System variable $AC_ESR_TRIGGER

### Programming

POLF[geo | mach]= value

**Target position of retracting axis**

### Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLF</td>
<td>Command, modal</td>
</tr>
<tr>
<td>geo</td>
<td>mach</td>
</tr>
<tr>
<td>value</td>
<td>Retract position, WCS is valid for geometry axis, otherwise MCS. When using the same identifiers for geometry axis and channel/machine axis, the workpiece coordinate system is used for retraction. Incremental programming is permissible.</td>
</tr>
</tbody>
</table>

### Programming

POLFMASK(axisname1, axisname2, ...)

**Axis selection for the retraction**

### Explanation of the commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLFMASK</td>
<td>Command</td>
</tr>
<tr>
<td>POLFMASK() without axis specification deactivates rapid lift for all axes.</td>
<td></td>
</tr>
<tr>
<td>axisname1</td>
<td>Names of the axes that are to travel to positions defined with POLF in case of LIFTFAST. All the axes specified must be in the same coordinate system. Before rapid lift to a defined position can be enabled via POLFMASK, you need to program a position via POLF for the selected axes.</td>
</tr>
</tbody>
</table>
There are no machine data with default settings for POLF values. When interpreting POLFMASK, alarm 16016 is issued if POLF has not yet been programmed.

**Notice**

The positions programmed with POLF and the activation via POLFMASK are deleted at parts program start. This means that the user must program the values for POLF and the selected axes (POLFMASK) in each parts program.

**Function**

**Stop**

The sequence for extended stop (NC-controlled) is specified in the following machine data:

- MD 21380: ESR_DELAY_TIME1
- MD 21381: ESR_DELAY_TIME2.

The axis continues interpolating as programmed for the time duration specified in MD 21380. After the time delay specified in MD 21380 has lapsed, controlled braking is initiated by interpolation. The maximum time available for the interpolatory controlled braking is specified in MD 21381; after this time has lapsed, rapid deceleration with subsequent correction is initiated.

The NC-controlled stop is

- configured: via MD 37500: 22 as well as 2 time specification via MD (see above);
- enabled ($AA_ESR_ENABLE) and
- triggered: System variable $AC_ESR_TRIGGER
13.6.3 Possible trigger sources

**Function**

The following error sources for starting "Extended stop and retract" are possible:

- **General sources (NC-external/global or mode group/channel-specific):**
  - Digital inputs (e.g. on NCU modules or terminal blocks) or mapping the digital outputs within the control ($A_IN, $A_OUT)
  - Channel status ($AC_STAT)
  - VDI signals ($A_DBB)
  - Group messages from a number of alarms ($AC_ALARM_STAT)

- **Axial sources:**
  - Emergency retraction threshold of the following axis (synchronization of electronic coupling, $VA_EG_SYNCDIFF[following axis])
  - Drive: DC link warning threshold (pending undervoltage), $AA_ESR_STAT[axis]
  - Drive: Generator minimum velocity threshold (no more regenerative rotation energy available), $AA_ESR_STAT[axis].

13.6.4 Logic gating functions: Source/reaction operation

**Function**

The static synchronized actions' flexible gating possibilities are used to trigger specific reactions according to the sources.

The operator has several options for gating all relevant sources by means of static synchronized actions. Users can evaluate the source system variable as a whole or also selectively by means of bit masks and gate their desired reactions to them. The static synchronized actions are effective in all operating modes.
13.6 Extended stopping and retract (SW 5 and higher)

For a more detailed description on how to use synchronized actions, please refer to
References: /FBSY/ Description of Functions
Synchronized Actions

13.6.5 Activation

Enabling functions:
$AA_ESR_ENABLE

The generator operation, stop and retract functions are enabled by setting the associated control signal ($AA_ESR_ENABLE). This control signal can be modified by the synchronized actions.

Triggering functions (general triggering of all released axes)
$AN_ESR_TRIGGER

- Generator operation is "automatically" active in the drive when a pending DC link undervoltage is detected.
- Drive-independent stop and/or retract are active when a communications failure (between the NC and drive) is detected, as well as when a DC link undervoltage is detected in the drive (providing it is configured and enabled).
- Drive-independent stop and/or retract can also be triggered from the NC side by setting the corresponding control signal $AN_ESR_TRIGGER (broadcast command to all drives).

13.6.6 Generator operation/DC link backup

Function

By configuring drive MD and carrying out the required programming via static synchronized actions ($AA_ESR_ENABLE), temporary DC link voltage drops can be compensated. The time that can be bridged depends on how much energy the generator that is used as DC link backup has stored, as well as how much energy is required to maintain the active movements (DC link backup and monitoring for generator speed limit).
When the value falls below the DC link voltage lower limit, the axis/spindle concerned switches from position or speed-controlled operation to generator operation. Drive deceleration (default speed setpoint = 0) causes regeneration of energy in the DC link. For more information see /FB/ M 3, Coupled Motion and Leading Value Coupling

13.6.7 Drive-independent stop

Function

The drives of a previously coupled grouping can be stopped by time-controlled cutout delay keeping the difference between them to a minimum, if the control is unable to achieve this. Drive-independent stop is configured and enabled via MD (delay time T1 in MD) and is enabled by system variable $AA_ESR_ENABLE and started with $AN_ESR_TRIGGER.

Reactions

For time T1 the speed setpoint that was active when the error occurred is still output. This is an attempt to maintain the movement that was active before the failure until the physical contact is annulled or the retraction movement initiated simultaneously in other drives is completed. This can be necessary for all leading/following drives or for drives that are coupled or in a grouping. After time T1, all axes with speed setpoint feedforward zero are stopped at the current limit, and the pulses are deleted when zero speed is reached or when the time has expired (+drive MD).
13.6.8 Drive-independent retract

Function

Axes with digital 611D drives can (if configured and released) also execute a retraction movement independently
• at control failure (sign-of-life detection)
• if the DC link voltage falls below a warning threshold
• if triggered by the system variable $AN_ESR_TRIGGER.

The retraction movement is performed independently by drive 611D.
Once the retraction phase is initiated, the drive independently maintains its enables at the values that were previously valid.

For more information see /FB/ M 3, Coupled Motion and Leading Value Coupling

13.6.9 Example: Using the drive-independent reaction

Example configuration

• Axis A is to operate as generator drive,
• axis X is to retract by 10 mm at maximum speed in event of an error and
• axes Y and Z are to stop with a time delay of 100 ms, such that the retraction axis has time to cancel the mechanical coupling.

Sequence

1. Activate options "Ext. Stop and retract" and "Mode-independent actions" (includes "Static synchronized actions IDS ...").
2. Function assignment:
   $MA_ESR_REACTION[X]=11,
   $MA_ESR_REACTION[Y]=12,
   $MA_ESR_REACTION[Z]=12,
   $MA_ESR_REACTION[A]=10;
Additional Functions

13.7 Link communication (SW 5.2 and higher)

3. Drive configuration:
   - MD 1639 RETRACT_SPEED[X] = 400000H in pos. direction (max. speed),
     = FFC00000H in neg. direction,
   - MD 1638 RETRACT_TIME[X] = 10ms (retract time),
   - MD 1637 GEN_STOP_DELAY[Y] = 100ms,
   - MD 1637 GEN_STOP_DELAY[Z] = 100ms,
   - MD 1635 GEN_AXIS_MIN_SPEED[A] = Generator min. speed (rpm).

4. Function enable (from parts program or synchronized actions):
   - $AA_ESR_ENABLE[X]=1,
   - $AA_ESR_ENABLE[Y]=1,
   - $AA_ESR_ENABLE[Z]=1,
   - $AA_ESR_ENABLE[A]=1

5. Get the generator operation to "momentum" speed
   (e.g. in spindle operation M03 S1000)

6. Formulate trigger condition as static synchronized action(s), e.g.:
   - dependent on intervention of the generator axis:
     IDS=01 WHENEVER $AA_ESR_STAT[A]>0 DO $AN_ESR_TRIGGER=1
   - and/or dependent on alarms that trigger follow-up mode
     (bit13=2000H):
     IDS=02 WHENEVER ($AC_ALARM_STAT B_AND 'H2000')>0
     DO $AN_ESR_TRIGGER=1
   - and also dependent on EU synchronized operation (if, for example, Y is defined as EU following axis and if the max. allowed deviation of synchronized operation shall be 100 µm):
     IDS=03 WHENEVER ABS($VA_EG_SYNCDIFF[Y])>0.1
     DO $AN_ESR_TRIGGER=1

13.7 Link communication (SW 5.2 and higher)

Function

The NCU link, which connects several NCU units from an installation, is used in configurations with a distributed system design. When there is a high demand for axes and channels, e.g. with revolving machines and multi-spindle machines, computing capacity, configuration options and memory areas can reach their limits when only one NCU is used.
Several networked NCUs connected by means of an NCU link module represent an open, scalable solution that meets all the requirements of this type of machine tool. The NCU link module (hardware) provides high-speed NCU-to-NCU communication.

Options providing this functionality can be ordered separately.

**Function**

Several NCUs linked via link modules can have read and write access to a global NCU memory area via the system variables described in the following.

- Each NCU linked via a link module can use **global link variables.** These link variables are addressed in the same way by all connected NCUs.
- Link variables can be programmed as system variables. As a rule, the machine manufacturer defines and documents the meaning of these variables.
- Applications for link variables:
  - Global machine states
  - Workpiece clamping open/closed
  - Etc.
- Relatively small data volume
- Very high transfer speed, therefore: Use is intended for time-critical information.
- These system variables can be accessed from the **parts program** and from **synchronized actions.** The size of the memory area for global NCU system variables configurable.

When a value is written in a global system variable, it can be read by all the NCUs connected after one interpolation cycle.
Link variables are **global system data** that can be addressed by the connected NCUs as **system variables**. The
- contents of these variables,
- their **data type**,
- **use**, and
- position (**access index**) in the link memory
are defined by the user (in this case generally the machine manufacturer).

Link variables are stored in the link memory.
After power-up, the link memory is initialized with 0.

The following link variables can be addressed within the link memory:
- **INT $A_DLB[i]** ; data byte (8 bits)
- **INT $A_DLW[i]** ; data word (16 bits)
- **INT $A_DLD[i]** ; double data word (32 bits)
- **REAL $A_DLR[i]** ; real data (64 bits)

According to the type in question, 1, 2, 4 or 8 bytes are addressed when the link variables are written/read.

Index i defines the start of the respective variable in relation to the start of the configured link memory.
The index is counted from 0 up.

**Value ranges**
The different data types have the following value ranges:

- **BYTE**: 0 to 255
- **WORD**: –32768 to 32767
- **DWORD**: –2147483648 to 2147483647
- **REAL**: –4.19e-308 to 4.19e-307

The various NCU applications sharing access to the link memory **at the same time** must use the link memory in a **uniform manner**. When the process is completely separate in time, the link memory can be occupied differently.
13.7 Link communication (SW 5.2 and higher)

Warning

A link variable write process is only then completed when the written information is also available to all the other NCUs. Approximately two interpolation cycles are necessary for this process. Local writing to the link memory is delayed by the same time for purposes of consistency.

For more information, please refer to the Description of Functions B3 (SW 5)

Programming example

$A\_DLB[5]=21$

The 5th byte in the shared link memory is assigned value 21.
13.8 Axis container (SW 5.2 and higher)

Function

With revolving machines/multi-spindle machines the axes holding the workpiece move from one machining station to the next. As the machining stations are controlled by different NCU channels, at station/position change the axes holding the workpiece must be dynamically reassigned to the appropriate NCU channel. The axis container is provided for this purpose. Only one workpiece clamping axis/spindle can be active at any one time at the local machining station. The axis container compiles the possible connections with all clamping axes/spindles, of which only exactly one is always activated for the machining station.

The following can be assigned via axis containers:
- Local axes and/or
- Link axes (see Fundamentals)

The available axes that are defined in the axis container can be changed by switching the entries in the axis container. This switching function can be triggered from the parts program.

The axis containers with link axes are a tool that is valid across NCUs (NCU global) and is coordinated by the control. It is also possible to have axis containers in which only local axes are managed.

Detailed information on configuring axis containers can be found in /FB/, B3 (SW 5.2)

The entries in the axis container can be switched by increment n via the commands:

Programming

AXCTSWE (CT,)  AXIS CONTAINER SWITCH ENABLE
AXCTSWED (CT,)  AXIS CONTAINER SWITCH ENABLE DIRECT
11.02 Additional Functions

13.8 Axis container (SW 5.2 and higher)

Explanation

CT_i or e.g. A_CONT1

Number of the axis container whose contents are to be switched or individual name of axis container set via MD.

Function

AXCTSWE()

Each channel whose axes are contained in the specified container issues an enable for a container rotation, if it has finished machining the position/station. Once the control receives the enables from all channels for the axes in the container, the container is rotated with the increment specified in the SD.

In the preceding example, after axis container rotation by 1, axis AX5 on NCU1 is assigned to channel axis Z instead of axis AX1 on NCU1.

The command variant AXCTSWED(CT_i) can be used to simplify startup. Under the sole effect of the active channel, the axis container rotates around the increment stored in the SD.
This call may only be used if the other channels, which have axes in the container are in the RESET state.

After an axis container rotation, all NCUs whose channels refer to the rotated axis container via the logical machine axis image are affected by the new axis assignment.

13.9 Program execution time/Workpiece counter (SW 5.2 and higher)

Function

Information on the program execution time and on the workpiece count are provided to support the person working at the machine tool. This information is specified in the respective machine data and can be edited as a system variable in the NC and/or PLC program. This information is also available to the MMC at the operator panel front interface.

13.9.1 Program runtime

Function

Under this function, timers are provided as system variables, which can be used to monitor technological processes. These timers can only be read. They can be accessed at any time by the MMC in read mode.

Explanation

The following two timers are defined as NCK-specific system variables and always active.

\[ \text{SAN\_SETUP\_TIME} \]

Time in minutes since the last setup; is reset with SETUP
### 13.9 Program execution time/Workpiece counter (SW 5.2 and higher)

The following three timers are defined as channel-specific system variables and can be activated via machine data:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$AN_POWERON_TIME</td>
<td>Time in minutes since the last PowerOn; is reset with POWERON</td>
</tr>
<tr>
<td>$$AC_OPERATING_TIME</td>
<td>Total execution time in seconds of NC programs in the automatic mode</td>
</tr>
<tr>
<td>$$AC_CYCLE_TIME</td>
<td>Execution time in seconds of the selected NC program</td>
</tr>
<tr>
<td>$$AC_CUTTING_TIME</td>
<td>Tool operation time in seconds</td>
</tr>
<tr>
<td>$$MC_RUNTIMER_MODE</td>
<td>Tool operation time in seconds</td>
</tr>
</tbody>
</table>

All timers are reset with default values when the control is powered up, and can be read independent of their activation.

**Programming example**

1. Activate runtime measurement for the active NC program; no measurement with active dry run feedrate and program testing:
   
   ```
   $\$MC\_PROCESSTIMER\_MODE = 'H2'
   ```

2. Activate measurement for the tool operating time; measurement also with active dry run feedrate and program testing:
   
   ```
   $\$MC\_PROCESSTIMER\_MODE= 'H34'
   ```

3. Activate measurement for the total runtime and tool operating time; measurement also during program testing:
   
   ```
   $\$MC\_PROCESSTIMER\_MODE= 'H25'
   ```
### 13.9.2 Workpiece counter

#### Function

The "Workpiece counter" function can be used to prepare counters, e.g. for internal counting of workpieces on the control. These counters exist as channel-specific system variables with read and write access within a value range from 0 to 999 999 999.

Machine data can be used to control counter activation, counter reset timing and the counting algorithm.

#### Explanation

The following counters are provided:

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC_REQUIRED_PARTS}$</td>
<td>Number of workpieces required. In this counter you can define the number of workpieces at which the actual workpiece counter $\text{AC_ACTUAL_PARTS}$ is reset to zero. Machine data can be used to configure the generation of the display alarm &quot;Required number of workpieces reached&quot; and the channel VDI signal &quot;Required number of workpieces reached&quot;.</td>
</tr>
<tr>
<td>$\text{AC_TOTAL_PARTS}$</td>
<td>Total number of workpieces actually produced (total actual). The counter indicates the total number of workpieces produced since the starting time. The counter is automatically reset with default values only when the control is powered up.</td>
</tr>
<tr>
<td>$\text{AC_ACTUAL_PARTS}$</td>
<td>Number of actual workpieces. This counter records the number of all workpieces produced since the starting time. The counter is automatically reset to zero (on condition that $\text{AC_REQUIRED_PARTS}$ is not equal to 0) when the required number of workpieces ($\text{AC_REQUIRED_PARTS}$) has been reached.</td>
</tr>
<tr>
<td>$\text{AC_SPECIAL_PARTS}$</td>
<td>Number of workpieces specified by the user. This counter allows user-defined workpiece counting. Alarm output can be defined for the case of identity with $\text{AC_REQUIRED_PARTS}$ (workpiece target). The user must reset the counter.</td>
</tr>
</tbody>
</table>

The "Workpiece counter" function operates independently of the tool management functions. All counters can be read and written from the MMC. All counters are reset with default values when the control is powered up, and can be read/written independent of their activation.
13.9 Program execution time/Workpiece counter (SW 5.2 and higher)

Programming example

1. Activate workpiece counter $AC_REQUIRED_PARTS:

   $MC_PART_COUNTER='H3'
   $AC_REQUIRED_PARTS is active, display
   alarm on $AC_REQUIRED_PARTS ==
   $AC_SPECIAL_PARTS

2. Activate workpiece counter $AC_TOTAL_PARTS:

   $MC_PART_COUNTER='H10'
   $MC_PART_COUNTER_MCODE[0]=80
   $AC_TOTAL_PARTS is active, the counter is
   incremented by 1 on each M02.
   $MC_PART_COUNTER_MCODE[0] is
   irrelevant

3. Activate workpiece counter $AC_ACTUAL_PARTS:

   $MC_PART_COUNTER='H300'
   $MC_PART_COUNTER_MCODE[1]=17
   $AC_TOTAL_PARTS is active, the counter is
   incremented by 1 on each M17

4. Activate workpiece counter $AC_SPECIAL_PARTS:

   $MC_PART_COUNTER='H3000'
   $MC_PART_COUNTER_MCODE[2]=77
   $AC_SPECIAL_PARTS is active, the counter
   is incremented by 1 on each M77

5. Deactivate workpiece counter $AC_ACTUAL_PARTS:

   $MC_PART_COUNTER='H200'
   $MC_PART_COUNTER_MCODE[1]=50
   $AC_TOTAL_PARTS is not active, rest
   irrelevant

6. Activate all counters, examples 1-4:

   $MC_PART_COUNTER = 'H3313'
   $AC_REQUIRED_PARTS is active
   Display alarm on $AC_REQUIRED_PARTS
   == $AC_SPECIAL_PARTS
   $AC_TOTAL_PARTS is active, the counter is
   incremented by 1 on each M02
   $MC_PART_COUNTER_MCODE[0] is
   irrelevant
   $AC_ACTUAL_PARTS is active, the counter
   is incremented by 1 on each M17
   $AC_SPECIAL_PARTS is active, the counter
   is incremented by 1 on each M77
13 Additional Functions

13.10 Interactive window call from parts program, command MMC

(SW 4.4 and higher)

Programming

MMC ("CYCLES, PICTURE_ON, T_SK.COM, PICTURE, MGUD.DEF, PICTURE_3.AWB, TEST_1, A1","S")

Explanation

CYCLES
Operating area in which the configured user dialog boxes are implemented.

PICTURE_ON or PICTURE_OFF
Command: display selection or display deselection

T_SK.COM
Com file: Name of the dialog display file (user cycles). The dialog display design is defined here. The dialog displays can show user variables and/or comments.

DISPLAY
Name of dialog display: The individual displays are selected via the names of the dialog displays.

MGUD.DEF
User data definition file, which is addressed while reading/writing variables.

PICTURE_3.AWB
Graphics file

TEST_1
Display time or acknowledgement variable

A1
"S"

Acknowledgement mode: synchronous, acknowledgement via "OK" soft key

Function

With the MMC command, user-defined dialog windows (dialog displays) can be displayed on the MMC/HMI from the parts program. The dialog window design is defined in pure text configuration (COM file in cycles directory), while the MMC/HMI system software remains unchanged. User-defined dialog windows cannot be called simultaneously in different channels.
Please see the detailed notes on how to program the MMC command (incl. programming examples) in `/IAM/` in the manuals IM1 through IM4 depending on the MMC/HMI software used.
13.11 Influencing the motion control

13.11.1 Percentage jerk correction: JERKLIM

Programming

JERKLIM[axis] = ...

Explanation of the command

<table>
<thead>
<tr>
<th>JERKLIM</th>
<th>Percentage change for the greatest permissible jerk relative to the value set in the machine data for the axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>Machine axis whose jerk limit has to adapted</td>
</tr>
</tbody>
</table>

Function

In critical program sections, it may be necessary to limit the jerk to below maximum value, for example, to reduce mechanical stress. The acceleration mode SOFT must be active. The function only effects path axes.

Sequence

In the AUTOMATIC modes, the jerk limit is limited to the percentage of the jerk limit stored in the machine data.

Example: N60 JERKLIM[X]=75
Meaning: The axis carriage in the X direction must be accelerated/decelerated with only 75% of the jerk permissible for the axis.

Value range: 1 ... 200
100 corresponds to: no effect on jerk.
100 is applied after RESET and parts program start.
13.11 Influencing the motion control

A further example will follow at the end of the next subsection.

13.11.2 Percentage velocity correction: VELOLIM

Programming

VELOLIM[axis]= ...

Explanation of the command

<table>
<thead>
<tr>
<th>VELOLIM</th>
<th>Percentage change for the greatest permissible velocity relative to the value set in the machine data for the axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>Machine axis whose velocity limit has to be adapted</td>
</tr>
</tbody>
</table>

Function

In critical program sections, it may be necessary to limit the velocity to below maximum values, for example, to reduce mechanical stress or enhance finish. The function only effects path and positioning axes.

Sequence

In the AUTOMATIC modes, the velocity limit is limited to the percentage of the velocity limit stored in the machine data.

Example: N70 VELOLIM[X]=80
Meaning: The axis carriage in the X direction must travel at only 80% of the velocity permissible for the axis.
### Additional Functions

#### 13.12 Master/slave grouping

**Value range:** 1 ... 100

- 100 corresponds to: no effect on velocity.
- 100 is applied after RESET and parts program start.

**Programming example**

```gcode
N1000 G0 X0 Y0 F10000 SOFT G64
N1100 G1 X20 RNDM=5 ACC[X]=20
               ACC[Y]=30
N1200 G1 Y20 VELOLIM[X]=5
               JERKLIM[Y]=200
N1300 G1 X0 JERKLIM[X]=2
N1400 G1 Y0
M30
```

#### 13.12 Master/slave grouping

**Programming:**

- `MASLDEF(Slv1, Slv2, ..., master axis)`
  - For dynamic configuration (SW 6.4 and higher)
- `MASLDEL(Slv1, Slv2, ..., )`
  - For dynamic configuration (SW 6.4 and higher)
- `MASLON(Slv1, Slv2, ..., )`
- `MASLOF(Slv1, Slv2, ..., )`
- `MASLOFS(Slv1, Slv2, ..., )`
  - (SW 6.4 and higher)

**Explanation of the parameters**

<table>
<thead>
<tr>
<th>Slv1, Slv2, ...</th>
<th>Slave axes led by a master axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master axis</td>
<td>Axis leading slave axes defined in a master/slave grouping</td>
</tr>
</tbody>
</table>
Function

The master/slave coupling in SW 6.4 and lower permitted coupling of the slave axes to their master axis only while the axes involved are stopped. Extension of SW 6.4 permits coupling and uncoupling of rotating, speed-controlled spindles and dynamic configuration.

Dynamic configuration

MASLDEF  
Definition of a master/slave grouping from the parts program. In SW 6.4 and lower, definition as in the machine data only.

MASLDEL  
The instruction cancels assignment of the slave axes to the master axis and simultaneously uncouples the current coupling, like MASLOF. The master/slave definitions declared in the machine data are retained.

General

MASLON  
Enable a temporary coupling

MASLOF  
This instruction uncouples an active coupling.  
For spindles in speed control mode, this instruction is executed immediately. The slave spindles rotating at this time retain their speeds until next speed programming.
**13.12 Master/slave grouping**

MASLOFS

(SW 6.4 and higher)
The MASLOFS instruction can be used to decelerate slave spindles automatically on uncoupling. For axes and spindles in positioning mode, uncoupling is only possible while stopped.

**More information (SW 6.4 and higher)**

For MASLOF/MASLOFS, the implicit preprocessing stop is not required. Because of the missing preprocessing stop, the $P$ system variables for the slave axes do not provide updated values until next programming.

**Programming example**

Dynamic configuration of a master/slave coupling from the parts program:

The axis relevant after axis container rotation must become the master axis.

```
MASLDEF (AUX, S3) ; S3 master for AUX
MASLON (AUX) ; Coupling ON for AUX
M3=3 S3=4000 ; Clockwise rotation
MASLDEL (AUX) ; Clear configuration and uncoupling
AXCTSWE (CT1) ; Container rotation
```
To enable coupling with another spindle after container rotation, the previous coupling must be uncoupled, the configuration cleared, and a new coupling configured.

Example of a coupling sequence Position 3 / Container CT1

See /FB/, B3 Section 2.6 Axis container
13.12 Master/slave grouping

840D
NCU 571

840D
NCU 572
NCU 573

810D

840Di
User Stock Removal Programs

14.1 Supporting functions for stock removal ................................................................. 14-542
14.2 Contour preparation: CONTPRON ..................................................................... 14-543
14.3 Contour decoding: CONTDCON (SW 5.2 and higher) ....................................... 14-550
14.4 Intersection of two contour elements: INTERSEC ........................................... 14-554
14.5 Traversing a contour element from the table: EXECTAB ................................. 14-556
14.6 Calculate circle data: CALCDAT ....................................................................... 14-557
14.1 Supporting functions for stock removal

User stock removal programs

Preprogrammed stock removal programs are provided for stock removal. You can also use the following functions to develop your own stock removal programs.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTPRON</td>
<td>Activate tabular contour preparation (11 columns)</td>
</tr>
<tr>
<td>CONTDCON</td>
<td>Activate tabular contour decoding (6 columns)</td>
</tr>
<tr>
<td>INTERSEC</td>
<td>Calculate intersection of two contour elements. (Only for tables created by CONTPRON).</td>
</tr>
<tr>
<td>EXECTAB</td>
<td>Block-by-block execution of contour elements of a table. (Only for tables created by CONTPRON).</td>
</tr>
<tr>
<td>CALCDAT</td>
<td>Calculate radii and center points</td>
</tr>
</tbody>
</table>

You can use these functions universally, not just for stock removal.
14.2 Contour preparation: CONTPRON

Programming

CONTPRON (TABNAME, MACH, NN, MODE)
EXECUTE (ERROR)

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTPRON</td>
<td>Activate contour preparation</td>
</tr>
<tr>
<td>TABNAME</td>
<td>Name of contour table</td>
</tr>
<tr>
<td>MACH</td>
<td>Parameters for type of machining:</td>
</tr>
<tr>
<td></td>
<td>&quot;G&quot;: Longitudinal turning: Inside machining</td>
</tr>
<tr>
<td></td>
<td>&quot;L&quot;: Longitudinal turning: External machining</td>
</tr>
<tr>
<td></td>
<td>&quot;N&quot;: Face turning: Inside machining</td>
</tr>
<tr>
<td></td>
<td>&quot;P&quot;: Face turning: External machining</td>
</tr>
<tr>
<td>NN</td>
<td>Number of relief cuts in result variable of type INT</td>
</tr>
<tr>
<td>MODE</td>
<td>Direction of machining, type INT</td>
</tr>
<tr>
<td></td>
<td>0 = Contour preparation forward (SW 4.3 and lower, default value)</td>
</tr>
<tr>
<td></td>
<td>1 = Contour preparation in both directions</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Terminate contour preparation</td>
</tr>
<tr>
<td>ERROR</td>
<td>Variable for error checkback, type INT</td>
</tr>
<tr>
<td></td>
<td>1 = error; 0 = no error</td>
</tr>
</tbody>
</table>

Function

The blocks executed after CONTPRON describe the contour to be prepared.
The blocks are not processed but are filed in the contour table.
Each contour element corresponds to one row in the two-dimensional array of the contour table.
The number of relief cuts is returned.
EXECUTE deactivates the contour preparation and switches back to the normal execution mode.

Example:
N30 CONTPRON(…)
N40 G1 X... Z...
N50...
N100 EXECUTE(…)

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition 14-543
Additional notes

Preconditions for the call
Before CONTPRON is called
- a starting point must be approached which permits collision-free machining,
- tool edge radius compensation with G40 must be deactivated.

Permitted traversing commands, coordinate system
Only G commands G0 to G3 are permitted for contour programming in addition to rounding and chamfer.
SW 4.4 and higher supports circular-path programming via CIP and CT.
The functions Spline, Polynomial, thread produce errors.
It is not permitted to change the coordinate system by activating a frame between CONTPRON and EXECUTE. The same applies to a change between G70 and G71/ G700 and G710.
Changing the geometry axes with GEOAX while preparing the contour table produced an alarm.

Terminate contour preparation
When you call the predefined subroutine EXECUTE (variable), contour preparation is terminated and the system switches back to normal execution when the contour has been described. The variable then indicates:
1 = error
0 = no error (the contour is error free).

Relief cut elements
The contour description for the individual relief cut elements can be performed either in a subroutine or in individual blocks.

Stock removal irrespective of the programmed contour direction (SW 4.4 and higher)
In SW 4.4 and higher, contour preparation has been expanded. Now when CONTPRON is called, the contour table is available irrespective of the programmed direction.
Programming example 1

Create a contour table with
- name KTAB,
- up to 30 contour elements (circles, straight lines),
- a variable for the number of relief cut elements,
- a variable for error messages

NC parts program

```
N10 DEF REAL KTAB[30,11]  Contour table named KTAB and, for
ex
ple, a maximum of 30 contour elements
Parameter value 11 is a fixed size

N20 DEF INT ANZHINT  Variable for number of relief cut elements
with name ANZHINT

N30 DEF INT ERROR  Variable for acknowledgment
0 = no error, 1 = error

N40 G18

N50 CONTPRON (KTAB,"G",ANZHINT)  Contour preparation call

N60 G1 X150 Z20  N60 to N120 contour description
N70 X110 Z30
N80 X50 RND=15
N90 Z70
N100 X40 Z85
N110 X30 Z90
N120 X0

N130 EXECUTE(ERROR)  Terminate filling of contour table, switch to
normal program execution

N140 ...  Continue processing table
```
### 14.2 Contour preparation: CONTPRON

#### Table KTAB

<table>
<thead>
<tr>
<th>(0)</th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
<th>(5)</th>
<th>(6)</th>
<th>(7)</th>
<th>(8)</th>
<th>(9)</th>
<th>(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>150</td>
<td>0</td>
<td>82.40535663</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>11</td>
<td>20</td>
<td>150</td>
<td>30</td>
<td>110</td>
<td>-</td>
<td>104.0362435</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1111</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>11</td>
<td>30</td>
<td>110</td>
<td>30</td>
<td>65</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>13</td>
<td>30</td>
<td>65</td>
<td>45</td>
<td>50</td>
<td>0</td>
<td>180</td>
<td>45</td>
<td>65</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>11</td>
<td>45</td>
<td>50</td>
<td>70</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>11</td>
<td>70</td>
<td>50</td>
<td>85</td>
<td>40</td>
<td>0</td>
<td>146.3099325</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>11</td>
<td>85</td>
<td>40</td>
<td>90</td>
<td>30</td>
<td>0</td>
<td>116.5650512</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>11</td>
<td>90</td>
<td>30</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Explanation of column contents

- **(0)** Pointer to next contour element (to the row number of that column)
- **(1)** Pointer to previous contour element
- **(2)** Coding of contour mode for the movement
  - Possible values for X = abc
    - a = 10^2, G90 = 0, G91 = 1
    - b = 10^1, G70 = 0, G71 = 1
    - c = 10^0, G0 = 0, G1 = 1, G2 = 2, G3 = 3
- **(3), (4)** Starting point of contour elements
  (3) = abscissa, (4) = ordinate in current plane
- **(5), (6)** Starting point of contour elements
  (5) = abscissa, (6) = ordinate in current plane
- **(7)** Max/min indicator: Identifies local maximum and minimum values on the contour
- **(8)** Maximum value between contour element and abscissa (for longitudinal machining) or ordinate (for transverse machining). The angle depends on the type of machining programmed.
- **(9), (10)** Center point coordinates of contour element, if it is a circle block.
  (9) = abscissa, (10) = ordinate
Programming example 2

Create a contour table with
- name KTAB,
- up to 92 contour elements (circles, straight lines),
- mode: Longitudinal turning, external machining
- preparation forwards and backwards.

NC parts program

```
N10 DEF REAL KTAB[92,11]          // Contour table named KTAB and, for example, a maximum of 92 contour elements
N20 CHAR BT="L"                     // Parameter value 11 is a fixed size
N30 DEF INT HE=0                    // Mode for CONTPRON: Longitudinal turning, external machining
N40 DEF INT MODE=1                  // Number of relief cut elements=0
N50 DEF INT ERR=0                   // Preparation forwards and backwards
...                                 // Error checkback message
N100 G18 X100 Z100 F1000           // Contour preparation call
N105 CONTPRON (KTAB, BT, HE, MODE)  // Terminate filling of contour table, switch to normal program execution
N110 G1 G90 Z20 X20                // Normal program execution
N120 X45                            // Normal program execution
N130 Z0                             // Normal program execution
N140 G2 Z-15 X30 K=AC(-15) I=AC(45) // Normal program execution
N150 G1 Z-30                         // Normal program execution
N160 X80                            // Normal program execution
N170 Z-40                           // Normal program execution
N180 EXECUTE(ERR)                   // Normal program execution
...                                 // Normal program execution
```
### Table KTAB

After contour preparation is finished, the contour is available in both directions.

<table>
<thead>
<tr>
<th>Row</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(0)</td>
</tr>
<tr>
<td>0</td>
<td>6(^1)</td>
</tr>
<tr>
<td>1</td>
<td>0(^3)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>1(^5)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>84</td>
</tr>
<tr>
<td>84</td>
<td>90</td>
</tr>
<tr>
<td>85</td>
<td>0(^8)</td>
</tr>
<tr>
<td>86</td>
<td>85</td>
</tr>
<tr>
<td>87</td>
<td>86</td>
</tr>
<tr>
<td>88</td>
<td>87</td>
</tr>
<tr>
<td>89</td>
<td>88</td>
</tr>
<tr>
<td>90</td>
<td>89</td>
</tr>
<tr>
<td>91</td>
<td>83(^9)</td>
</tr>
</tbody>
</table>

**Explanation of column contents**

- **(0)** Pointer to next contour element (to the row number of that column)
- **(1)** Pointer to previous contour element
- **(2)** Coding of contour mode for the movement
  - Possible values for \( X = abc \)
    - \( a = 10^2 \)  \( G90 = 0 \)  \( G91 = 1 \)
    - \( b = 10^1 \)  \( G70 = 0 \)  \( G71 = 1 \)
    - \( c = 10^0 \)  \( G0 = 0 \)  \( G1 = 1 \)  \( G2 = 2 \)  \( G3 = 3 \)
- **(3), (4)** Starting point of contour elements
  - (3) = abscissa, (4) = ordinate in current plane
- **(5), (6)** Starting point of contour elements
  - (5) = abscissa, (6) = ordinate in current plane
- **(7)** Max/min indicator: Identifies local maximum and minimum values on the contour
- **(8)** Maximum value between contour element and abscissa (for longitudinal machining) or ordinate (for transverse machining)
- The angle depends on the type of machining programmed.
(9), (10) Center point coordinates of contour element, if it is a circle block.
(9) = abscissa, (10) = ordinate

Explanation of comment in columns

*Always in table line 0:*
1) Previous: Line n contains the contour end forwards
2) Following: Line n is the contour table end forwards

*Once each within the contour elements forwards:*
3) Previous: Contour start (forwards)
4) Following: Contour end (forwards)

*Always in line contour table end (forwards) +1:*
5) Previous: Number of relief cuts forwards
6) Following: Number of relief cuts backwards

*Once each within the contour elements backwards:*
7) Following: Contour end (backwards)
8) Previous: Contour start (backwards)

*Always in last line of table:*
9) Previous: Line n is the contour table start (backwards)
10) Following: Line n contains the contour start (backwards)
14.3 Contour decoding: CONTDCON (SW 5.2 and higher)

**Programming**

```
CONTDCON (TABNAME, MODE)
EXECUTE (ERROR)
```

**Explanation of the parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTDCON</td>
<td>Activate contour preparation</td>
</tr>
<tr>
<td>TABNAME</td>
<td>Name of contour table</td>
</tr>
<tr>
<td>MODE</td>
<td>Direction of machining, type INT</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Terminate contour preparation</td>
</tr>
<tr>
<td>ERROR</td>
<td>Variable for error checkback, type INT</td>
</tr>
</tbody>
</table>

0 = Contour preparation (default) according to the contour block sequence

1 = error; 0 = no error

**Function**

The blocks executed after CONTPRON describe the contour to be decoded.

The blocks are not processed but stored, memory-optimized, in a 6-column contour table.

Each contour element corresponds to one row in the contour table. When familiar with the coding rules specified below, you can combine DIN code programs from the tables to produce applications (e.g. cycles). The data for the starting point are stored in the table cell with the number 0. The G codes permitted for CONTDCON in the program section to be included in the table are more comprehensive than for the CONTPRON function. In addition, feedrates and feed type are also stored for each contour section.

EXECUTE deactivates the contour preparation and switches back to the normal execution mode.

**Example:**

```
N30  CONTDCON(...)
N40  G1 X... Z...
N50...
N100  EXECUTE(...)
```
Additional notes

Preconditions for the call
Before CONTDCON is called
- a starting point must be approached which permits collision-free machining,
- tool edge radius compensation with G40 must be deactivated.

Permitted traversing commands, coordinate system
The following G groups and specified commands are permissible for contour programming:
G group 1: G0, G1, G2, G3
G group 10: G9
G group 11: G60, G44, G641, G642
G group 13: G70, G71, G700, G710
G group 14: G90, G91
G group 15: G93, G94, G95, G96
also corner and chamfer.
Circular-path programming is possible via CIP and CT. The functions Spline, Polynomial, thread produce errors.

It is not permitted to change the coordinate system by activating a frame between CONDCRON and EXECUTE. The same applies to a change between G70 and G71/ G700 and G710. Changing the geometry axes with GEOAX while preparing the contour table produced an alarm.
Terminate contour preparation
When you call the predefined subroutine EXECUTE (ERROR), contour preparation is terminated and the system switches back to normal execution when the contour has been described. The associated variable ERROR gives the return value:
0 = no error (contour produced no errors)
1 = error
Impermissible commands, incorrect initial conditions, CONTDCON call repeated without EXECUTE( ), too few contour blocks or table definitions too small produce additional alarms.

Stock removal in the programmed contour direction
The contour table produced using CONTDCON is used for stock removal in the programmed direction of the contour.

Programming example
Create a contour table with
• name KTAB,
• contour elements (circles, straight lines),
• mode: Turning
• preparation forward
### NC parts program

**N10** DEF REAL KTAB[9, 6]  
Contour table with name KTAB and 9 table cells. These allow 8 contour sets. Parameter value 6 (column number in table) is fixed.

**N20** DEF INT MODE = 0  
Default value 0: Only in programmed contour direction. Value 1 is not permitted.

**N30** DEF INT ERROR = 0  
Error checkback message

...  
**N100** G18 G64 G90 G94 G710  
**N101** G1 Z100 X100 F1000  
**N105** CONTDON (KTAB, MODE)  
Call contour decoding  
MODE may be omitted (see above)

**N110** G1 Z20 X20 F200  
**N120** G9 X45 F300  
**N130** Z0 F400  
**N140** G2 Z-15 X30 K=AC(-15) I=AC(45)F100  
**N150** G64 Z-30 F600  
**N160** X80 F700  
**N170** Z-40 F800  
**N180** EXECUTE (ERROR)  
Terminate filling of contour table, switch to normal program execution

...  

<table>
<thead>
<tr>
<th>Column index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line index</td>
<td>Contour mode</td>
<td>End point abscissa</td>
<td>End point ordinate</td>
<td>Center point Abscissa</td>
<td>Center point ordinate</td>
<td>Feed</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>11031</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>111031</td>
<td>20</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>300</td>
</tr>
<tr>
<td>3</td>
<td>11031</td>
<td>0</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>400</td>
</tr>
<tr>
<td>4</td>
<td>11032</td>
<td>-15</td>
<td>30</td>
<td>-15</td>
<td>45</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>11031</td>
<td>-30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>600</td>
</tr>
<tr>
<td>6</td>
<td>11031</td>
<td>-30</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>700</td>
</tr>
<tr>
<td>7</td>
<td>11031</td>
<td>-40</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>800</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
14.4 Intersection of two contour elements: INTERSEC

Programming

VARIB=INTERSEC (TABNAME1[n1], TABNAME2[n2], TABNAME3)

Explanation of the parameters

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIB</td>
<td>Variable for status, TRUE: Intersection found, FALSE: No intersection found</td>
</tr>
<tr>
<td>TABNAME1[n1]</td>
<td>Table name and n1st contour element of the first table</td>
</tr>
<tr>
<td>TABNAME2[n2]</td>
<td>Table name and n2nd contour element of the second table</td>
</tr>
<tr>
<td>TABNAME3</td>
<td>Table name for the intersection coordinates in the active plane G17–G19</td>
</tr>
</tbody>
</table>

Explanation of column contents

Line 0: Coding for starting point:

- Column 0:
  - $10^0$ (ones): $G0 = 0$
  - $10^1$ (tens): $G70 = 0, G71 = 1, G700 = 2, G710 = 3$

- Column 1: starting point of abscissa
- Column 2: starting point of ordinate
- Column 3-4: 0
- Column 5: Line index of last contour piece in the table

Lines 1-n: Entries for contour pieces:

- Column 0:
  - $10^0$ (ones): $G0 = 0, G1 = 1, G2 = 2, G3 = 3$
  - $10^1$ (tens): $G70 = 0, G71 = 1, G700 = 2, G710 = 3$
  - $10^2$ (hundreds): $G90 = 0, G91 = 1$
  - $10^3$ (thousands): $G93 = 0, G94 = 1, G95 = 2, G96 = 3$
  - $10^4$ (ten thousands): $G60 = 0, G44 = 1, G641 = 2, G642 = 3$
  - $10^5$ (hundred thousands): $G9 = 1$

- Column 1: End point abscissa
- Column 2: End point ordinate
- Column 3: Center point Abscissa for circular interpolation
- Column 4: Center point ordinate for circular interpolation
- Column 5: Feedrate
14.4 Intersection of two contour elements: INTERSEC

Function
INTERSEC calculates the intersection of two normalized contour elements from the contour table generated with CONTPRON. The indicated status specifies whether or not an intersection exists (TRUE = intersection, FALSE = no intersection).

Additional notes
Please note that variables must be defined before they are used.

Programming example
Calculate the intersection of contour element 3 in table KTAB1 and contour element 7 in table KTAB2. The intersection coordinates in the active plane are stored in CUT (1st element = abscissa, 2nd element = ordinate).
If no intersection exists, the program jumps to NOCUT (no intersection found).

```plaintext
DEF REAL KTAB1 [12, 11] Contour table 1
DEF REAL KTAB2 [10, 11] Contour table 2
DEF REAL CUT [2] Intersection table
DEF BOOL ISPOINT Variable for status
...
N10 ISPOINT=INTERSEC (KTAB1[3],KTAB2[7],CUT) Call intersection of contour elements
N20 IF ISPOINT==FALSE GOTOF NOCUT Jump to NOCUT
...
```
14.5 Traversing a contour element from the table: EXECTAB

**Programming**

`EXECTAB (TABNAME[n])`

**Explanation of the parameter**

<table>
<thead>
<tr>
<th>TABNAME[n]</th>
<th>Name of table with number n of the element</th>
</tr>
</thead>
</table>

**Function**

You can use command EXECTAB to traverse contour elements block by block in a table generated, for example, with the CONTPRON command.

**Programming example**

The contour elements stored in Table KTAB are traversed non-modally by means of subroutine EXECTAB. Elements 0 to 2 are passed in consecutive calls.

N10 `EXECTAB (KTAB[0])`  Traverse element 0 of table KTAB
N20 `EXECTAB (KTAB[1])`  Traverse element 1 of table KTAB
N30 `EXECTAB (KTAB[2])`  Traverse element 2 of table KTAB
14.6 Calculate circle data: CALCDAT

Programming

```
VARIB = CALCDAT(PT[n,2],NO,RES)
```

Explanation of the parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| VARIB     | Variable for status  
TRUE = circle, FALSE = no circle |
| PT[n,2]   | Points for calculation  
n = number of points (3 or 4); 2 = point coordinates |
| NO.       | Number of points used for calculation: 3 or 4 |
| RES[3]    | Variable for result: specification of circle center point coordinates and radius;  
0 = abscissa, 1 = ordinate of circle center point; 2 = radius |

Function

Calculation of radius and circle center point coordinates from three or four known circle points.  
The specified points must be different.  
Where 4 points do not lie directly on the circle an average value is taken for the circle center point and the radius.
Programming example

The program determines whether the three points lie along the arc of a circle.

```
N10  DEF REAL
    PT[3,2]=(20,50,50,40,65,20)  # Point definition
N20  DEF REAL RES[3]
N30  DEF BOOL STATUS
N40  STATUS = CALCDAT(PT,3,RES)  # Call calculated circle data
N50  IF STATUS == FALSE GOTOF ERROR  # Jump to error
```

Diagram:

- Three points: (20,50), (50,40), (65,20)
- Determining if they lie along the arc of a circle.
## Tables

15.1 List of instructions ................................................................. 15-561

15.2 List of system variables .......................................................... 15-591
15.2.1 R parameters ................................................................. 15-591
15.2.2 Channel-specific synchronized action variables ................. 15-591
15.2.3 Frames 1 ........................................................................ 15-592
15.2.4 Toolholder data ............................................................... 15-593
15.2.5 Channel-specific protection zones ..................................... 15-601
15.2.6 Tool parameters ............................................................... 15-603
15.2.7 Cutting edge data OEM user ............................................ 15-609
15.2.8 Monitoring data for tool management ................................ 15-617
15.2.9 Monitoring data for OEM users ....................................... 15-618
15.2.10 Tool-related data ............................................................ 15-619
15.2.11 Tool-related grinding data ............................................. 15-621
15.2.12 Magazine location data ................................................ 15-622
15.2.13 Magazine location data for OEM users ......................... 15-623
15.2.14 Magazine description data for tool management ............ 15-624
15.2.15 Tool management magazine description data for OEM users 15-625
15.2.16 Magazine module parameter .......................................... 15-626
15.2.17 Adapter data ................................................................. 15-626
15.2.18 Measuring system compensation values ......................... 15-626
15.2.19 Quadrant error compensation ........................................ 15-627
15.2.20 Interpolatory compensation ........................................... 15-629
15.2.21 NCK-specific protection zones ....................................... 15-630
15.2.22 Cycle parameterization ................................................ 15-631
15.2.23 System data .................................................................. 15-636
15.2.24 Frames 2 ...................................................................... 15-636
15.2.25 Tool data ...................................................................... 15-638
15.2.26 Magazines ................................................................... 15-643
15.2.27 Programmed geometry axis values ................................ 15-646
15.2.28 G groups ..................................................................... 15-647
15.2.29 Programmed values ...................................................... 15-647
15.2.30 Channel states ............................................................... 15-651
15.2.31 Synchronized actions ................................................... 15-656
15.2.32 I/Os ............................................................................. 15-657
15.2.33 Reading and writing PLC variables ............................... 15-657
15.2.34 NCU link ..................................................................... 15-658
15.2.35 Direct PLC I/O .............................................................. 15-658
15.2.36 Tool management .......................................................... 15-659
15.2.37 Timers ........................................................................ 15-662
15.2.38 Path movement .............................................................. 15-663
15.2.39 Speeds/accelerations ................................................... 15-665
| 15.2.40 | Spindles | 15-667 |
| 15.2.41 | Polynomial values for synchronized actions | 15-670 |
| 15.2.42 | Channel states | 15-672 |
| 15.2.43 | Measurement | 15-673 |
| 15.2.44 | Positions | 15-677 |
| 15.2.45 | Indexing axes | 15-679 |
| 15.2.46 | Encoder values | 15-679 |
| 15.2.47 | Axial measurement | 15-680 |
| 15.2.48 | Offsets | 15-681 |
| 15.2.49 | Axial paths | 15-684 |
| 15.2.50 | Oscillation | 15-685 |
| 15.2.51 | Axial velocities | 15-685 |
| 15.2.52 | Drive data | 15-687 |
| 15.2.53 | Axis statuses | 15-688 |
| 15.2.54 | Master/slave links | 15-689 |
| 15.2.55 | Travel to fixed stop | 15-690 |
| 15.2.56 | Electronic gear | 15-691 |
| 15.2.57 | Leading value coupling | 15-692 |
| 15.2.58 | Synchronized spindle | 15-693 |
| 15.2.59 | Safety Integrated | 15-696 |
### 15.1 List of instructions

#### Legend:

1. Default setting at start of program (in delivery state of control system provided that another setting is not programmed).
2. The group numbers correspond to the table "List of G functions/Preparatory functions" in /PG/, Programming Guide Fundamentals, Section 12.3
3. Absolute end points: Modal; incremental end points: Non-modal; otherwise modal/non-modal depending on syntax of G function
4. IPO parameters act incrementally as arc centers. They can be programmed in absolute mode with AC. When they have other meanings (e.g. pitch), the address modification is ignored.
5. Vocabulary word does not apply to SINUMERIK FM-NC/810D
6. Vocabulary word does not apply to SINUMERIK FM-NC/810D/NCU571
7. Vocabulary word does not apply to SINUMERIK 810D
8. The OEM user can incorporate two extra interpolation types and modify their names.
9. Vocabulary word applies only to SINUMERIK FM-NC
10. The extended address block format may not be used for these functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Value assignment</th>
<th>Description, comment</th>
<th>Syntax</th>
<th>Modal/ non-modal</th>
<th>Group²</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>Block number - main block (see N)</td>
<td>0 ... 9999 9999 integer values only, no sign</td>
<td>Special code for blocks - instead of N... ; this block should contain all instructions for a following complete machining section</td>
<td>e.g. :20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Axis</td>
<td>Real</td>
<td>m,s³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2 ⁵</td>
<td>Tool orientation: Euler angle</td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3 ⁵</td>
<td>Tool orientation: Direction vector component</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4 ⁵</td>
<td>Tool orientation for block beginning</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5 ⁵</td>
<td>Tool orientation for block end; Normal vector component</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABS</td>
<td>Absolute value</td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>Dimension input, absolute</td>
<td>0, ..., 359.9999°</td>
<td>X=AC(100)</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACC ⁵</td>
<td>Axial acceleration</td>
<td>Real, without sign</td>
<td>m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACN</td>
<td>Absolute dimension setting for rotary axes, approach position in negative direction</td>
<td>A=ACN(...) B=ACN(...) C=ACN(…)</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macro</td>
<td>Description</td>
<td>Type</td>
<td>Arguments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACP</td>
<td>Absolute dimension setting for rotary axes, approach position in positive direction</td>
<td>Real, without sign</td>
<td>A=ACP(...) B=ACP(...) C=ACP(...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACOS</td>
<td>Arc cosine (trigon. function)</td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADIS</td>
<td>Resurfacing distance for path functions G1, G2, G3, ...</td>
<td>Real, without sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADISPOS</td>
<td>Resurfacing distance for rapid traverse G0</td>
<td>Real, without sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADISPOSA</td>
<td>Size of the tolerance window for IPOBRKA</td>
<td>Integer, real, without sign</td>
<td>ADISPOSA= or ADISPOSA(&lt;axis&gt;,REAL)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALF</td>
<td>Angle lift fast</td>
<td>Integer, without sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMIRROR</td>
<td>Programmable mirroring (additive mirror)</td>
<td></td>
<td>AMIRROR X0 Y0 Z0 ; separate block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>Logical AND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANG</td>
<td>Contour definition angle</td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP</td>
<td>Polar angle (Angle Polar)</td>
<td>0, ..., ± 360°</td>
<td>m,s³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APR</td>
<td>Read/display access protection (access protection read)</td>
<td>Integer, without sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APW</td>
<td>Write access protection (access protection write)</td>
<td>Integer, without sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>Aperture angle (angle circular)</td>
<td>0, ..., 360°</td>
<td>m,s³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AROT</td>
<td>Programmable rotation (additive rotation)</td>
<td>Rotation around 1st geom. axis: -180° .. 180°, 2nd geom. axis: -89.999° .. 90°, 3rd geo. axis: -180° .. 180°</td>
<td>AROT X... Y... Z... ; separate block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AROTS</td>
<td>Programmable frame rotations with solid angles (additive rotation)</td>
<td></td>
<td>AROT X... Y... Z... ; own block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS</td>
<td>Macro definition</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASCALE</td>
<td>Programmable scaling (additive scale)</td>
<td></td>
<td>ASCALE X... Y... Z... ; separate block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASIN</td>
<td>Arc sine (trigon. function)</td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASPLINE</td>
<td>Akima spline</td>
<td></td>
<td>m 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAN2</td>
<td>Arc tangent 2</td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATRANS</td>
<td>Additive programmable offset (additive translation)</td>
<td></td>
<td>ATRANS X... Y... Z... ; separate block</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AX</td>
<td>Integer without sign</td>
<td>Real</td>
<td>m,s³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Data Type</td>
<td>Units</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AXCSWAP</strong></td>
<td>Switch container axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AXIS</strong></td>
<td>Data type: Axis name</td>
<td>Name of file can be added</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AXNAME</strong></td>
<td>Converts the input string to an axis name (get axname)</td>
<td>String</td>
<td>An alarm is generated if the input string does not contain a valid axis name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>AXSTRING</strong></td>
<td>Up to SW 5, axis identifier is converted to string (get axis as string) With SW 6 and higher, the spindle number converts the string (get string)</td>
<td>Up to SW 5 AXIS from SW 6 string</td>
<td>Name of file can be added</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>Axis</td>
<td>Real</td>
<td>m, s³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B_AND</strong></td>
<td>Bit AND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B_NOT</strong></td>
<td>Bit negation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B_OR</strong></td>
<td>Bit OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B_XOR</strong></td>
<td>Bit exclusive OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B2</strong></td>
<td>Tool orientation: Euler angle</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B3</strong></td>
<td>Tool orientation: Direction vector component</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B4</strong></td>
<td>Tool orientation for block beginning</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B5</strong></td>
<td>Tool orientation for block end; Normal vector component</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BAUTO</strong></td>
<td>Definition of first spline segment by means of following 3 points (begin not a knot)</td>
<td>m</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BLSYNC</strong></td>
<td>Processing of interrupt routine is only to start with the next block change</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BNAT</strong></td>
<td>Natural transition to first spline block (begin natural)</td>
<td>m</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BOOL</strong></td>
<td>Data type: Boolean value TRUE / FALSE or 0 / 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BRISK</strong></td>
<td>Brisk path acceleration</td>
<td>m</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BRISKA</strong></td>
<td>Activate brisk axis acceleration for the programmed axes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BSPLINE</strong></td>
<td>B spline</td>
<td>m</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BTAN</strong></td>
<td>Tangential transition to first spline block (begin tangential)</td>
<td>m</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>Axis</td>
<td>Real</td>
<td>m, s³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C2</strong></td>
<td>Tool orientation: Euler angle</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C3</strong></td>
<td>Tool orientation: Direction vector component</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C4</strong></td>
<td>Tool orientation for block beginning</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Type</td>
<td>Parameter(s)</td>
<td>Notes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>------</td>
<td>--------------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>Tool orientation for block end; Normal vector component</td>
<td>Real</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAC</td>
<td>Absolute approach of position (coded position: absolute coordinate)</td>
<td>Coded value is table index; table value is approached</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CACN</td>
<td>Absolute approach in negative direction of value stored in table. (coded position absolute negative)</td>
<td>Permissible for programming rotary axes as positioning axes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CACP</td>
<td>Absolute approach in positive direction of value stored in table. (coded position absolute positive)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALCDATA</td>
<td>Calculate radius and center point or circle from 3 or 4 points (calculate circle data)</td>
<td>VAR Real [3]</td>
<td>The points must be different.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>Indirect subroutine call</td>
<td>CALL PROGVAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALLPATH</td>
<td>Programmable search path for subprogram calls</td>
<td>A path can be programmed to the existing NCK file system with CALLPATH.</td>
<td>CALLPATH(/N_WKS_DIR/N_MYWPD/subprogram_ID_SPF)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CANCEL</td>
<td>Cancel modal synchronized action</td>
<td>INT</td>
<td>Cancel with specified ID. Without parameter: All modal synchronized actions are deselected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASE</td>
<td>Conditional program branch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDC</td>
<td>Direct approach of position (coded position: direct coordinate)</td>
<td>See CAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDOF</td>
<td>Collision detection OFF</td>
<td>m</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDON</td>
<td>Collision detection ON</td>
<td>m</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDOF2</td>
<td>Collision detection OFF</td>
<td>For CUT3DC only</td>
<td>m</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFC</td>
<td>Constant feed on contour</td>
<td>m</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFIN</td>
<td>Constant feed at internal radius only, not at external radius</td>
<td>m</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CFTCP</td>
<td>Constant feed at tool center point (center-point path) (constant feed in tool-center-point)</td>
<td>m</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAN</td>
<td>Specify validity range for data</td>
<td>once per channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHANDATA</td>
<td>Set channel number for channel data access</td>
<td>INT</td>
<td>Only permissible in the initialization module</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------</td>
<td>-----</td>
<td>--------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>Data type: ASCII character</td>
<td>0, ..., 255</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHF</td>
<td>Chamfer; value = length of chamfer in direction of movement (chamfer)</td>
<td>Real, without sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chamfer; value = length of chamfer</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHKMDNO</td>
<td>D number check</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIC</td>
<td>Incremental approach of position (coded position: incremental coordinate)</td>
<td>See CAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIP</td>
<td>Circular interpolation through intermediate points</td>
<td>CIP X... Y... Z... I1=... J1=... K1=...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEARM</td>
<td>Reset one/several markers for channel coordination</td>
<td>INT, 1 - n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Does not influence machining in own channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLGOF</td>
<td>Const. workpiece speed for centerless grinding OFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLGON</td>
<td>Const. workpiece speed for centerless grinding ON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLRINT</td>
<td>Deselect interrupt: INT Parameter: Interrupt number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMIRROR</td>
<td>Mirror on a coordinate axis FRAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COARSEA</td>
<td>Motion end when &quot;Exact stop coarse&quot; reached</td>
<td>COARSEA=.. or COARSEA[n]=..</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPOF</td>
<td>Compressor OFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>m 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPON</td>
<td>Compressor ON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>m 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPCURV</td>
<td>Compressor ON constant curve polynomials</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>m 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPCAD</td>
<td>Compressor ON optimized surface finish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>m 30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTRPON</td>
<td>Activate contour preparation (contour preparation ON)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>m 49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>Cosine (trigon. function)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUPEDEF</td>
<td>Definition ELG group / synchronous spindle group (couple definition)</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Block change (software) response: NOC: no software control, FINE/COARSE: software on &quot;Synchronization fine / coarse&quot;, IPOSTOP: software on setpoint-dependent termination of overlaid movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUPDEL</td>
<td>Delete ELG group (couple delete)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUPOF</td>
<td>ELG group / synchronous spindle pair OFF (couple OFF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUPON</td>
<td>ELG group / synchronous spindle pair ON (couple ON)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.1 List of instructions

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUPRE</td>
<td>Reset ELG group (couple reset)</td>
<td>Programmed values invalid; machine data values valid</td>
</tr>
<tr>
<td>CP</td>
<td>Path movement (continuous path)</td>
<td>m 49</td>
</tr>
<tr>
<td>CPRECOF</td>
<td>Programmable contour precision OFF</td>
<td>m 39</td>
</tr>
<tr>
<td>CPRECON</td>
<td>Programmable contour precision ON</td>
<td>m 39</td>
</tr>
<tr>
<td>CPROT</td>
<td>Channel-specific protection zone ON/OFF</td>
<td>m 39</td>
</tr>
<tr>
<td>CPROTDEF</td>
<td>Channel specific protection area definition</td>
<td></td>
</tr>
<tr>
<td>CR</td>
<td>Circle radius</td>
<td>Real, without sign</td>
</tr>
<tr>
<td>CROT</td>
<td>Rotation of the current coordinate system.</td>
<td>FRAME Maximum number of parameters: 6</td>
</tr>
<tr>
<td>CROTS</td>
<td>Programmable frame rotations with solid angles (rotations in the indicated axes)</td>
<td>CROT X... Y... CROT Z... X... CROT Y... Z... ; own CROT RPL= block s</td>
</tr>
<tr>
<td>CScale</td>
<td>Scale factor for multiple axes.</td>
<td>FRAME Maximum number of parameters: 2 * axis number max</td>
</tr>
<tr>
<td>CSpline</td>
<td>Cubic spline</td>
<td></td>
</tr>
<tr>
<td>CTAB</td>
<td>Define following axis position according to leading axis position from curve table</td>
<td>Real If parameter 4/5 not programmed: Standard scaling</td>
</tr>
<tr>
<td>CTABDEF</td>
<td>Table definition ON</td>
<td></td>
</tr>
<tr>
<td>CTABDEL</td>
<td>Clear curve table</td>
<td></td>
</tr>
<tr>
<td>CTABEND</td>
<td>Table definition OFF</td>
<td></td>
</tr>
<tr>
<td>CTABINV</td>
<td>Define leading axis position according to following axis position from curve table</td>
<td>Real See CTAB</td>
</tr>
<tr>
<td>CT</td>
<td>Circle with tangential transition</td>
<td>CT X... Y... Z... m 1</td>
</tr>
<tr>
<td>CTRANS</td>
<td>Zero offset for multiple axes</td>
<td>FRAME Max. of 8 axes</td>
</tr>
<tr>
<td>CUT2D</td>
<td>Tool offset (cutter compensation type 2-dimensional)</td>
<td>m 22</td>
</tr>
<tr>
<td>CUT2DF</td>
<td>Tool offset (cutter compensation type 2-dimensional frame); The tool offset acts in relation to the current frame (inclined plane)</td>
<td>m 22</td>
</tr>
<tr>
<td>CUT3DC</td>
<td>3D cutter compensation type 3-dimensional circumference milling</td>
<td>m 22</td>
</tr>
<tr>
<td>CUT3DCC</td>
<td>Cutter compensation type 3-dimensional circumference milling with limit surfaces</td>
<td>m 22</td>
</tr>
</tbody>
</table>
### 15.1 List of instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CUT3DCCD</strong></td>
<td>Cutter compensation type 3-dimensional circumference milling with differential tool</td>
</tr>
<tr>
<td><strong>CUT3DF</strong></td>
<td>3D tool offset face milling (cutter compensation type 3-dimensional face)</td>
</tr>
<tr>
<td><strong>CUT3DFF</strong></td>
<td>3D tool offset face milling with constant tool orientation as a function of active frame (cutter compensation type 3-dimensional face frame)</td>
</tr>
<tr>
<td><strong>CUT3DFS</strong></td>
<td>3D tool offset face milling with constant tool orientation irrespective of active frame (cutter compensation type 3-dimensional face frame)</td>
</tr>
<tr>
<td><strong>CUTCONO</strong></td>
<td>Constant radius compensation OFF</td>
</tr>
<tr>
<td><strong>CUTCONON</strong></td>
<td>Constant radius compensation ON</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>Tool offset number</td>
</tr>
<tr>
<td><strong>DC</strong></td>
<td>Absolute dimension setting for rotary axes, approach position directly</td>
</tr>
<tr>
<td><strong>DEF</strong></td>
<td>Variable definition</td>
</tr>
<tr>
<td><strong>DEFAULT</strong></td>
<td>Branch in CASE branch</td>
</tr>
<tr>
<td><strong>DEFINE</strong></td>
<td>Define macro</td>
</tr>
<tr>
<td><strong>DELDTG</strong></td>
<td>Delete distance-to-go</td>
</tr>
<tr>
<td><strong>DELT</strong></td>
<td>Delete tool</td>
</tr>
<tr>
<td><strong>DIAMCYOF</strong></td>
<td>Radius programming for G90/91: ON. The G-code of this group that was last active remains active for display</td>
</tr>
<tr>
<td><strong>DIAMOF</strong></td>
<td>Diametral programming: OFF</td>
</tr>
<tr>
<td><strong>DIAMON</strong></td>
<td>Diametral programming: ON</td>
</tr>
<tr>
<td><strong>DIAM90</strong></td>
<td>Diameter program for G90, radius progr. for G91</td>
</tr>
<tr>
<td><strong>DILF</strong></td>
<td>Rapid lift length</td>
</tr>
<tr>
<td><strong>DISABLE</strong></td>
<td>Interrupt OFF</td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISC</td>
<td>Transition circle overshoot in tool radius compensation</td>
<td>0, ..., 100</td>
<td>m</td>
</tr>
<tr>
<td>DISPLOF</td>
<td>Suppress current block display (display OFF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPR</td>
<td>Distance path for repositioning</td>
<td>Real, without sign</td>
<td>s</td>
</tr>
<tr>
<td>DISR</td>
<td>Distance for repositioning</td>
<td>Real, without sign</td>
<td>s</td>
</tr>
<tr>
<td>DITE</td>
<td>Thread run-out path</td>
<td>Real</td>
<td>m</td>
</tr>
<tr>
<td>DITS</td>
<td>Thread run-in path</td>
<td>Real</td>
<td>m</td>
</tr>
<tr>
<td>DIV</td>
<td>Integer division</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DL</td>
<td>Tool sum compensation</td>
<td>INT</td>
<td>m</td>
</tr>
<tr>
<td>DRFOF</td>
<td>Deactivate the handwheel offsets (DRF)</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>DRIVE⁹</td>
<td>Velocity-dependent path acceleration</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>DRIVEA</td>
<td>Switch on bent acceleration characteristic curve for the programmed axes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DZERO</td>
<td>Set D number of all tools of the TO unit assigned to the channel invalid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAUTO</td>
<td>Definition of last spline segment by last 3 points (end not a knot)</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>EGDEF</td>
<td>Definition of an electronic gear (Electronic gear define)</td>
<td>for 1 following axis with up to 5 leading axes</td>
<td></td>
</tr>
<tr>
<td>EGDEL</td>
<td>Delete coupling definition for the following axis (Electronic gear delete)</td>
<td>Triggers preprocessing stop</td>
<td></td>
</tr>
<tr>
<td>EGOFC</td>
<td>Switch off electronic gear continuous (Electronic gear OFF continuous)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGOFS</td>
<td>Switch off electronic gear selectively (Electronic gear OFF selective)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGON</td>
<td>Switch on electronic gear (electronic gear ON)</td>
<td>without synchronization</td>
<td></td>
</tr>
<tr>
<td>EGONSYN</td>
<td>Switch on electronic gear (electronic gear ON synchronized)</td>
<td>with synchronization</td>
<td></td>
</tr>
<tr>
<td>EGONSYN</td>
<td>Switch on electronic gearing, stating approach mode (electronic gear ON synchronized)</td>
<td>with synchronization</td>
<td></td>
</tr>
<tr>
<td>ELSE</td>
<td>Program branch, if IF condition not fulfilled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENABLE</td>
<td>Interrupt ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENAT¹,²*</td>
<td>Natural curve transition to next traversing block (end natural)</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>ENDFOR</td>
<td>End line of FOR counter loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDIF</td>
<td>End line of IF branch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDLOOP</td>
<td>End line of endless program loop LOOP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Type</td>
<td>Example</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>ENDPROC</td>
<td>End line of program with start line PROC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDWHILE</td>
<td>End line of WHILE loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETAN</td>
<td>Tangential curve transition to next traversing block at beginning of spline (end tangential)</td>
<td>m</td>
<td>20</td>
</tr>
<tr>
<td>EVERY</td>
<td>Execute synchronized action if condition changes from FALSE to TRUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXECSTRING</td>
<td>Transfer of a string variable with the parts program line to run</td>
<td>Indirect parts program line</td>
<td>EXECSTRING(MFCT1 &lt;&lt; M4711)</td>
</tr>
<tr>
<td>EXECTAB</td>
<td>Execute an element from a motion table (execute table)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Program execution ON</td>
<td>Switch back to normal program execution from reference point edit mode or after creating a protection zone</td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td>Exponent function $e^x$</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>EXTCALL</td>
<td>Run external subprogram</td>
<td>Reload program from HMI in &quot;Processing from external source&quot; mode</td>
<td></td>
</tr>
<tr>
<td>EXTERN</td>
<td>Broadcast a subroutine with parameter passing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Feed value (dwell time is also programmed under F in conjunction with G4)</td>
<td>0.001, ..., 99 999.999</td>
<td>Tool/workpiece path velocity; Dimension in mm/min or mm/revolution as a function of G94 or G95</td>
</tr>
<tr>
<td>FA</td>
<td>Axial feed (feed axial)</td>
<td>0.001, ..., 99999.999</td>
<td>mm/min, degree/min; 0.001, ..., 39999.9999 inch/min</td>
</tr>
<tr>
<td>FAD</td>
<td>Infeed feedrate for smooth approach and retraction (Feed approach / depart)</td>
<td>Real, without sign</td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td>Logical constant: False</td>
<td>BOOL</td>
<td>Can be replaced with integer constant 0</td>
</tr>
<tr>
<td><strong>Instruction</strong></td>
<td><strong>Description</strong></td>
<td><strong>M/s</strong></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>FCTDEF</td>
<td>Define polynomial function</td>
<td>Is evaluated in SYFCT or PUTFTOCF.</td>
<td></td>
</tr>
<tr>
<td>FCB</td>
<td>Feed variable according to cubic spline (feed cubic)</td>
<td>m 37</td>
<td></td>
</tr>
<tr>
<td>FD</td>
<td>Path feed for handwheel override (feed DRF)</td>
<td>Real, without sign s</td>
<td></td>
</tr>
<tr>
<td>FDA</td>
<td>Axial feed for handwheel override (feed DRF axial)</td>
<td>Real, without sign s</td>
<td></td>
</tr>
<tr>
<td>FENDNORM</td>
<td>Corner deceleration OFF</td>
<td>m 57</td>
<td></td>
</tr>
<tr>
<td>FFWOF</td>
<td>Feedforward control OFF (feed forward OFF)</td>
<td>m 24</td>
<td></td>
</tr>
<tr>
<td>FFWON</td>
<td>Feedforward control ON (feed forward ON)</td>
<td>m 24</td>
<td></td>
</tr>
<tr>
<td>FGROUP</td>
<td>Define axis(axes) with path feed</td>
<td>F applies to all axes programmed under FGROUP FGROUP (Axis1, [Axis2], ...)</td>
<td></td>
</tr>
<tr>
<td>FIFOCTRL</td>
<td>Control of the preprocessing memory</td>
<td>m 4</td>
<td></td>
</tr>
<tr>
<td>FIFOLEN</td>
<td>Programmable preprocessing depth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINEA</td>
<td>Motion end when &quot;Exact stop fine” reached</td>
<td>FINEA=... or FINEA[n]=... m</td>
<td></td>
</tr>
<tr>
<td>FL</td>
<td>Limit velocity for synchronous axes (feed limit)</td>
<td>Real, without sign The unit set with G93, G94, G95 applies (max. rapid traverse) FL [Axis] =... m</td>
<td></td>
</tr>
<tr>
<td>FLIN</td>
<td>Linearly variable feed (feed linear)</td>
<td>m 37</td>
<td></td>
</tr>
<tr>
<td>FMA</td>
<td>Feed multiple axial</td>
<td>Real, without sign m</td>
<td></td>
</tr>
<tr>
<td>FNORM</td>
<td>Normal feed acc. to DIN66025 (feed normal)</td>
<td>m 37</td>
<td></td>
</tr>
<tr>
<td>FOR</td>
<td>Counter loop with fixed number of passes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOR1</td>
<td>Feed for swiveling the orientation vector on the large circle</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>FOR2</td>
<td>Feed for the overlaid rotation around the swiveled orientation vector</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>Fixed point: numb. of fixed points to be approached</td>
<td>Integer, without sign G75 FP=1 s</td>
<td></td>
</tr>
<tr>
<td>FPO</td>
<td>Feed characteristic programmed via a polynomial (feed polynomial)</td>
<td>Real Quadratic, cubic polynomial coefficient</td>
<td></td>
</tr>
<tr>
<td>FPR</td>
<td>Rotary axis identification</td>
<td>0.001... 999999.999 FPR (rotary axis)</td>
<td></td>
</tr>
<tr>
<td>FPRAOF</td>
<td>Deactivate revolutional feedrate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPRAON</td>
<td>Activate revolutional feedrate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td>Data type to define the coordinate system</td>
<td>Contains for each geometry axis: Offset, rotation, angle of shear, scaling, mirroring; For each special axis: Offset, scaling, mirroring</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>FRC</td>
<td>Feed for radius and chamfer</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>FRCM</td>
<td>Feed for radius and chamfer modal</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>FTOC</td>
<td>Change fine tool offset</td>
<td>As a function of a 3rd degree polynomial defined with FCTDEF</td>
<td></td>
</tr>
<tr>
<td>FTOCOF</td>
<td>Online fine tool offset OFF</td>
<td>m 33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(fine tool offset OFF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTOCON</td>
<td>Online fine tool offset ON</td>
<td>m 33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(fine tool offset ON)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FXS</td>
<td>Travel to fixed stop ON (fixed stop)</td>
<td>Integer, without sign 1 = select, 0 = deselect</td>
<td></td>
</tr>
<tr>
<td>FXST</td>
<td>Torque limit for travel to fixed stop</td>
<td>% Optional setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(fixed stop torque)</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>FXSW</td>
<td>Monitoring window for travel to fixed</td>
<td>mm, inch or degree Optional setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stop (fixed stop window)</td>
<td>m</td>
<td></td>
</tr>
</tbody>
</table>
## G functions

<table>
<thead>
<tr>
<th>G</th>
<th>G function (preparatory function)</th>
<th>Integer, preset values only</th>
<th>G...</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>G functions are divided into G groups. Only one of the G functions in a group may be programmed in a block. A G function can be modally active (until it is canceled by another function in the same group) or it is active only in the block in which it is programmed (non-modal).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| G0 | Linear interpolation with rapid traverse (rapid traverse motion) | Motion | G0 X... Z... m 1 |
| G1 | Linear interpolation with feed (linear interpolation) | commands | G1 X... Z... F... m 1 |

| G2 | Circular interpolation clockwise | G2 X... Z... I... K... F... ; center and end points | m 1 |
|    |                                 | G2 X... Z... CR=... F... ; radius and end points | |
|    |                                 | G2 AR=... I... K... F... ; aperture angle and center point | |
|    |                                 | G2 AR=... X... Z... F... ; aperture angle and end point | |

| G3 | Circular interpolation counterclockwise | G3 ... ; otherwise as for G2 | m 1 |

| G4 | Predefined dwell time | Special motion | G4 F... ; dwell time in s or G4 S... ; dwell time in spindle rotations ; separate block | s 2 |

| G9 | Exact stop deceleration | | s 11 |

| G17 | Selection of working plane X/Y | Infeed direction Z | m 6 |
| G18 | Selection of working plane Z/X | Infeed direction Y | m 6 |
| G19 | Selection of working plane Y/Z | Infeed direction X | m 6 |

| G25 | Lower working area limitation | Value assignment in channel axes | G25 X... Y... Z... ; separate block | s 3 |

<p>| G26 | Upper working area limitation | G26 X... Y... Z... ; separate block | s 3 |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>G33</strong></td>
<td>Thread interpolation with constant pitch</td>
<td>0.001,..., 2000.00 mm/rev</td>
<td>Motion command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G34</strong></td>
<td>Increase in thread pitch (progressive change)</td>
<td></td>
<td>Motion command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G35</strong></td>
<td>Decrease in thread pitch (degressive change)</td>
<td></td>
<td>Motion command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G40</strong></td>
<td>Tool radius compensation OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G41</strong></td>
<td>Tool radius compensation to left of contour</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G42</strong></td>
<td>Tool radius compensation to right of contour</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G53</strong></td>
<td>Suppression of current zero offset (non-modal)</td>
<td></td>
<td>incl. Programmed offsets</td>
</tr>
<tr>
<td><strong>G54</strong></td>
<td>1st settable zero offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G55</strong></td>
<td>2nd settable zero offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G56</strong></td>
<td>3rd settable zero offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G57</strong></td>
<td>4th settable zero offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G58</strong></td>
<td>Programmable offset replacing axially</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G59</strong></td>
<td>Programmable offset replacing additive axially</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G60</strong></td>
<td>Exact stop deceleration</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G62</strong></td>
<td>Corner deceleration at inside corners with active tool radius compensation (G41, G42)</td>
<td></td>
<td>Together with continuous-path mode only</td>
</tr>
<tr>
<td><strong>G63</strong></td>
<td>Tapping with compensating chuck</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G64</strong></td>
<td>Exact stop - contouring mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G70</strong></td>
<td>Dimension in inches (lengths)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G71</strong></td>
<td>Metric dimension (lengths)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G74</strong></td>
<td>Reference point approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G75</strong></td>
<td>Fixed point approach</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G90</strong></td>
<td>Dimension setting, absolute</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G91</strong></td>
<td>Incremental dimension setting</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
15-573
### 15.1 List of instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Parameters</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>G93</td>
<td>Inverse-time feedrate rpm</td>
<td>Execution of a block: Time</td>
<td>G93 G01 X... F... m 15</td>
</tr>
<tr>
<td>G94</td>
<td>Linear feed F in mm/min or inch/min and °/min</td>
<td></td>
<td>m 15</td>
</tr>
<tr>
<td>G95</td>
<td>Revolutionary feedrate F in mm/rev or inch/rev</td>
<td></td>
<td>m 15</td>
</tr>
<tr>
<td>G96</td>
<td>Constant cutting speed ON</td>
<td>G96 S... LIMS=... F... m 15</td>
<td></td>
</tr>
<tr>
<td>G97</td>
<td>Constant cutting speed OFF</td>
<td></td>
<td>m 15</td>
</tr>
<tr>
<td>G110</td>
<td>Polar programming relative to last programmed set position</td>
<td>G110 X.. Y.. Z..</td>
<td>s 3</td>
</tr>
<tr>
<td>G111</td>
<td>Pole programming relative to zero point of current workpiece coordinate system</td>
<td>G110 X.. Y.. Z..</td>
<td>s 3</td>
</tr>
<tr>
<td>G112</td>
<td>Polar programming relative to last valid pole</td>
<td>G110 X.. Y.. Z..</td>
<td>s 3</td>
</tr>
<tr>
<td>G140</td>
<td>Direction of approach WAB defined by G41/G42</td>
<td></td>
<td>m 43</td>
</tr>
<tr>
<td>G141</td>
<td>Direction of approach WAB left of contour</td>
<td></td>
<td>m 43</td>
</tr>
<tr>
<td>G142</td>
<td>Direction of approach WAB right of contour</td>
<td></td>
<td>m 43</td>
</tr>
<tr>
<td>G143</td>
<td>Direction of approach WAB dependent on tangent</td>
<td></td>
<td>m 43</td>
</tr>
<tr>
<td>G147</td>
<td>Smooth approach with straight line</td>
<td>s 2</td>
<td></td>
</tr>
<tr>
<td>G148</td>
<td>Smooth retraction with straight line</td>
<td>s 2</td>
<td></td>
</tr>
<tr>
<td>G153</td>
<td>Suppression of current frame incl. base frame</td>
<td>s 9</td>
<td></td>
</tr>
<tr>
<td>G247</td>
<td>Smooth approach with quadrant</td>
<td>s 2</td>
<td></td>
</tr>
<tr>
<td>G248</td>
<td>Smooth retraction with quadrant</td>
<td>s 2</td>
<td></td>
</tr>
<tr>
<td>G331</td>
<td>Tapping</td>
<td>± 0.001, ..., 2000.00 mm/rev, Motion commands</td>
<td>m 1</td>
</tr>
<tr>
<td>G332</td>
<td>Retraction (tapping)</td>
<td></td>
<td>m 1</td>
</tr>
<tr>
<td>G340</td>
<td>Approach block spatial (depth and in plane at same time (helix))</td>
<td>for smooth approach and retract</td>
<td>m 44</td>
</tr>
<tr>
<td>G341</td>
<td>Approach in the perpendicular axis (z), then approach in plane</td>
<td>for smooth approach and retract</td>
<td>m 44</td>
</tr>
<tr>
<td>G347</td>
<td>Smooth approach with semicircle</td>
<td>s 2</td>
<td></td>
</tr>
<tr>
<td>G348</td>
<td>Smooth retract with semi-circle</td>
<td>s 2</td>
<td></td>
</tr>
<tr>
<td>G450</td>
<td>Transition circle</td>
<td>Tool compensation response at corners</td>
<td>m 18</td>
</tr>
<tr>
<td>G451</td>
<td>Intersection of equidistant paths</td>
<td></td>
<td>m 18</td>
</tr>
<tr>
<td>G460</td>
<td>Approach/retraction behavior with TRC</td>
<td></td>
<td>m 48</td>
</tr>
<tr>
<td>G461</td>
<td>Approach/retraction behavior with TRC</td>
<td></td>
<td>m 48</td>
</tr>
<tr>
<td>G462</td>
<td>Approach/retraction behavior with TRC</td>
<td></td>
<td>m 48</td>
</tr>
<tr>
<td>G500</td>
<td>Deactivation of all settable frames, if no value in G500</td>
<td></td>
<td>m 8</td>
</tr>
<tr>
<td>G505 - G599</td>
<td>5. ... 99. Settable zero offset</td>
<td></td>
<td>m 8</td>
</tr>
<tr>
<td>G601</td>
<td>Block change in response to exact stop fine</td>
<td>Effective only in conjunction with active G60 or G9</td>
<td>m</td>
</tr>
<tr>
<td>G602</td>
<td>Block change in response to exact stop coarse</td>
<td>m</td>
<td>12</td>
</tr>
<tr>
<td>G603</td>
<td>Block change in response to IPO end of block</td>
<td>m</td>
<td>12</td>
</tr>
<tr>
<td>G641</td>
<td>Exact stop - contouring mode</td>
<td>with programmable transition rounding</td>
<td>G641 ADIS=...</td>
</tr>
<tr>
<td>G642</td>
<td>Rounding with axial precision</td>
<td>m</td>
<td>10</td>
</tr>
<tr>
<td>G643</td>
<td>Block-internal corner rounding</td>
<td>m</td>
<td>10</td>
</tr>
<tr>
<td>G644</td>
<td>Smoothing with axis dynamics default</td>
<td>m</td>
<td>10</td>
</tr>
<tr>
<td>G621</td>
<td>Corner deceleration at all corners</td>
<td>Together with continuous-path mode only</td>
<td>G621 ADIS=...</td>
</tr>
<tr>
<td>G700</td>
<td>Dimension in inches and inch/min (lengths + velocities + system variable)</td>
<td>m</td>
<td>13</td>
</tr>
<tr>
<td>G710*</td>
<td>Metric dimension in mm and mm/min (lengths + velocities + system variable)</td>
<td>m</td>
<td>13</td>
</tr>
<tr>
<td>G810*, ..., G819</td>
<td>G group reserved for OEM users</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>G820*, ..., G829</td>
<td>G group reserved for OEM users</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>G931</td>
<td>Feedrate specified by travel time</td>
<td>Travel time</td>
<td>m</td>
</tr>
<tr>
<td>G942</td>
<td>Freeze linear feedrate and constant cutting rate or spindle speed</td>
<td>m</td>
<td>15</td>
</tr>
<tr>
<td>G952</td>
<td>Freeze revolutional feedrate and constant cutting rate or spindle speed</td>
<td>m</td>
<td>15</td>
</tr>
<tr>
<td>G961</td>
<td>Constant cutting speed ON</td>
<td>without additional spindle rotation</td>
<td>G961 S... LIMS=... F...</td>
</tr>
<tr>
<td>G962</td>
<td>Linear or revolutional feedrate and constant cutting rate</td>
<td>m</td>
<td>15</td>
</tr>
<tr>
<td>G971</td>
<td>Constant cutting speed OFF</td>
<td>m</td>
<td>15</td>
</tr>
<tr>
<td>G972</td>
<td>Freeze linear or revolutional feedrate and constant spindle speed</td>
<td>m</td>
<td>15</td>
</tr>
<tr>
<td>GEOAX</td>
<td>Assign new channel axes to geometry axes 1 - 3</td>
<td>Without parameter: MD settings effective</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>Assign machine axis/axes</td>
<td>Axis must be released in the other channel with RELEASE</td>
<td></td>
</tr>
<tr>
<td>GETD</td>
<td>Assign machine axis/axes directly</td>
<td>See GET</td>
<td></td>
</tr>
<tr>
<td>GETACTT</td>
<td>Get active tool from a group of tools with the same name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETSELT</td>
<td>Get selected T number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETT</td>
<td>Get T number for tool name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTOF</td>
<td>Jump instruction forwards (towards the end of program)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTOB</td>
<td>Jump instruction back (towards start of program)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 15.1 List of instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOTO</td>
<td>Jump instruction first forward then backward (direction initially to end of program and then to start of program)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTO</td>
<td>Alarm 14080 Suppress jump destination not found. See GOTO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GWPSOF</td>
<td>Deselect constant grinding wheel peripheral speed (GWPS)</td>
<td>GWPSOF (T No.)</td>
<td>s</td>
</tr>
<tr>
<td>GWPSON</td>
<td>Select constant grinding wheel peripheral speed (GWPS)</td>
<td>GWPSON (T No.)</td>
<td>s</td>
</tr>
<tr>
<td>H...</td>
<td>Auxiliary function output to PLC</td>
<td>Real/INT</td>
<td>Settable via MD (machine manufacturer) = H100 or H2=100</td>
</tr>
<tr>
<td>I^4</td>
<td>Interpolation parameter</td>
<td>Real</td>
<td>s</td>
</tr>
<tr>
<td>I1</td>
<td>Intermediate point coordinate</td>
<td>Real</td>
<td>s</td>
</tr>
<tr>
<td>IC</td>
<td>Incremental dimension setting</td>
<td>0, ..., ±99999.999</td>
<td>X=IC(10) = s</td>
</tr>
<tr>
<td>IDS</td>
<td>Identification of static synchronized actions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>Introduce conditional jump</td>
<td></td>
<td>Structure: IF - ELSE - ENDIF</td>
</tr>
<tr>
<td>INDEX</td>
<td>Define index of character in input string</td>
<td>0, ..., INT</td>
<td>String: Param. 1, character: Param. 2</td>
</tr>
<tr>
<td>INIT</td>
<td>Select module for execution in a channel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>Data type: Integer with leading sign</td>
<td>2^{31-1}, ..., 2^{31-1}</td>
<td></td>
</tr>
<tr>
<td>INTERSEC</td>
<td>Calculate intersection between two contour elements</td>
<td>VAR REAL [2]</td>
<td>Error status BOOL</td>
</tr>
<tr>
<td>IP</td>
<td>Variable interpolation parameter</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>IPOBRKA</td>
<td>Motion criterion from braking ramp activation</td>
<td>Braking ramp with 100% to 0%</td>
<td>IPOBRKA=.. or IPOBRKA(&lt;axis&gt;[,REAL]) = m</td>
</tr>
<tr>
<td>IPOENDA</td>
<td>Motion end when &quot;IPO stop&quot; reached</td>
<td></td>
<td>IPOENDA=.. or IPOENDA[n].. = m</td>
</tr>
<tr>
<td>ISAXIS</td>
<td>Check if geometry axis 1 – 3 specified as parameter exist</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>ISD</td>
<td>Insertion depth</td>
<td>Real</td>
<td>m</td>
</tr>
<tr>
<td>ISNUMBER</td>
<td>Check whether the input string can be converted to a number</td>
<td>BOOL</td>
<td>Convert input string to number</td>
</tr>
<tr>
<td>ISVAR</td>
<td>Check whether the transfer parameter contains a variable known in the NC</td>
<td>BOOL</td>
<td>Machine data, setting data and variables as GUDs</td>
</tr>
<tr>
<td>J^4</td>
<td>Interpolation parameter</td>
<td>Real</td>
<td>s</td>
</tr>
<tr>
<td>J1</td>
<td>Intermediate point coordinate</td>
<td>Real</td>
<td>s</td>
</tr>
<tr>
<td>JERKA</td>
<td>Activate acceleration response set via machine data for programmed axes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K^4</td>
<td>Interpolation parameter</td>
<td>Real</td>
<td>s</td>
</tr>
<tr>
<td>K1</td>
<td>Intermediate point coordinate</td>
<td>Real</td>
<td>s</td>
</tr>
<tr>
<td>KONT</td>
<td>Traverse around contour for tool compensation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Subprogram number Integer, up to 7 places</td>
<td>L10</td>
<td>s</td>
</tr>
<tr>
<td>LEAD 6</td>
<td>Lead angle Real</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEADOF</td>
<td>Leading value coupling OFF (lead off)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEADON</td>
<td>Leading value coupling ON (lead on)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LFOF 1</td>
<td>Interruption of thread cutting OFF m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LFON</td>
<td>Interruption of thread cutting ON m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LFTXT 1</td>
<td>Tool direction tangential at lift</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LFWP</td>
<td>Tool direction not tangential at lift</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIFTFAST</td>
<td>Rapid lift before interrupt routine call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIMS</td>
<td>Spindle speed limitation (limit spindle speed) with G96</td>
<td>0.001 ... 99 999.999</td>
<td>m</td>
</tr>
<tr>
<td>LN</td>
<td>Natural logarithm Real</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOCK</td>
<td>Disable synchronized action with ID (stop technology cycle)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>(Common) logarithm Real</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOOP</td>
<td>Introduction of an endless loop</td>
<td>Structure: LOOP - ENDOLOOP</td>
<td></td>
</tr>
<tr>
<td>M...</td>
<td>Switching operations 0, ..., 9999 9999 Max. of 5 free special functions to be defined by machine manufacturer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M0 10</td>
<td>Programmed stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M1 10</td>
<td>Optional stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2 10</td>
<td>Program end, main program with reset to program start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M3</td>
<td>Clockwise spindle rotation for master spindle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M4</td>
<td>Counterclockwise spindle rotation for master spindle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td>Spindle stop for master spindle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M6</td>
<td>Tool change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M17 10</td>
<td>End of subprogram</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M19</td>
<td>Spindle positions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M30 10</td>
<td>Program end, as for M2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M40</td>
<td>Automatic gear change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M41... M45</td>
<td>Gear stage 1, ..., 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M70</td>
<td>Transition to axis operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASLDEF</td>
<td>Define master/slave axis grouping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASLDEL</td>
<td>Uncouple master/slave axis grouping and clear grouping definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASLOF</td>
<td>Disable a temporary coupling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Data Type</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>MASLOFS</td>
<td>Deactivate a temporary coupling with automatic slave axis stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASLON</td>
<td>Enable a temporary coupling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCALL</td>
<td>Modal subprogram call</td>
<td>Without subprogram name: Deselection</td>
<td></td>
</tr>
<tr>
<td>MEAC</td>
<td>Continuous measurement without deletion of distance-to-go</td>
<td>Integer, without sign</td>
<td>s</td>
</tr>
<tr>
<td>MEAFRAME</td>
<td>Frame calculation from measuring points</td>
<td>FRAME</td>
<td></td>
</tr>
<tr>
<td>MEAS</td>
<td>Measurement with touch trigger probe (measure)</td>
<td>Integer, without sign</td>
<td>s</td>
</tr>
<tr>
<td>MEASA</td>
<td>Measurement with deletion of distance-to-go</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>MEAW</td>
<td>Measurement with touch trigger probe without deletion of distance-to-go</td>
<td>Integer, without sign</td>
<td>s</td>
</tr>
<tr>
<td>MEAWA</td>
<td>Measurement without deletion of distance-to-go</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>MI</td>
<td>Access to frame data: Mirroring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINDEX</td>
<td>Define index of character in input string</td>
<td>0, ..., INT</td>
<td>String: Parameter 1, character: Parameter 2</td>
</tr>
<tr>
<td>MIRROR</td>
<td>Programmable mirror</td>
<td>MIRROR X0 Y0 Z0</td>
<td>s 3</td>
</tr>
<tr>
<td>MMC</td>
<td>Calling the dialog window interactively from the parts program on the MMC/HMI</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>MOD</td>
<td>Modulo division</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>Start positioning axis (start moving positioning axis)</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>MSG</td>
<td>Programmable messages</td>
<td>MSG(&quot;message&quot;)</td>
<td>m</td>
</tr>
<tr>
<td>N</td>
<td>Subblock number</td>
<td>0, ..., 9999 9999 integer values only, no sign</td>
<td>Can be used to identify blocks with a number; position at beginning of block</td>
</tr>
<tr>
<td>NCK</td>
<td>Specify validity range for data</td>
<td></td>
<td>once per NCK</td>
</tr>
<tr>
<td>NEWCONF</td>
<td>Accept modified machine data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEWT</td>
<td>Create new tool</td>
<td>Duplo number can be omitted</td>
<td></td>
</tr>
<tr>
<td>NORM</td>
<td>Normal setting at start and end points for tool offset</td>
<td>m 17</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>Logical NOT (negation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPROT</td>
<td>Machine-specific protection zone ON/OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPROTDEF</td>
<td>Machine-specific protection area definition (NCK-specific protection area definition)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>NUMBER</td>
<td>Convert input string to number</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>OEMIPO1,6,8</td>
<td>OEM interpolation 1</td>
<td>m 1</td>
<td></td>
</tr>
<tr>
<td>OEMIPO2,6,8</td>
<td>OEM interpolation 2</td>
<td>m 1</td>
<td></td>
</tr>
<tr>
<td>OF</td>
<td>Vocabulary word in CASE branch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OFFN</td>
<td>Allowance for programmed contour</td>
<td>OFFN=5</td>
<td></td>
</tr>
<tr>
<td>OMA1 6</td>
<td>OEM address 1</td>
<td>Real m</td>
<td></td>
</tr>
<tr>
<td>OMA2 6</td>
<td>OEM address 2</td>
<td>Real m</td>
<td></td>
</tr>
<tr>
<td>OMA3 6</td>
<td>OEM address 3</td>
<td>Real m</td>
<td></td>
</tr>
<tr>
<td>OMA4 6</td>
<td>OEM address 4</td>
<td>Real m</td>
<td></td>
</tr>
<tr>
<td>OMA5 6</td>
<td>OEM address 5</td>
<td>Real m</td>
<td></td>
</tr>
<tr>
<td>OFFN</td>
<td>Offset compensation - normal</td>
<td>Real m</td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td>Logical OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIC1,6</td>
<td>Changes in orientation at outer corners are overlaid on the circular block to be inserted (orientation change continuously)</td>
<td>m 27</td>
<td></td>
</tr>
<tr>
<td>ORID6</td>
<td>Changes in orientation are performed before the circular block (orientation change discontinuously)</td>
<td>m 27</td>
<td></td>
</tr>
<tr>
<td>ORIAPOS</td>
<td>Orientation angle via virtual orientation axes with rotary axis positions</td>
<td>m 50</td>
<td></td>
</tr>
<tr>
<td>ORIEULER</td>
<td>Orientation angles using Euler angles</td>
<td>m 50</td>
<td></td>
</tr>
<tr>
<td>ORIAxes</td>
<td>Linear interpolation of machine axes or orientation axes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORICONCW</td>
<td>Interpolation on a circular peripheral surface in CW direction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORICONCCW</td>
<td>Interpolation on a circular peripheral surface in CCW direction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORICONIO</td>
<td>Interpolation on a circular peripheral surface with intermediate orientation setting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORICONTO</td>
<td>Interpolation on circular peripheral surface in tangential transition (final orientation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORICURVE</td>
<td>Interpolation of orientation with specification of motion of two contact points of tool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIPLANE</td>
<td>Interpolation in a plane (corresponds to ORIVECT)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIPATH</td>
<td>Tool orientation trajectory referred to path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIRPY</td>
<td>Orientation angles using RPY angles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIS</td>
<td>Change in orientation (orientation smoothing factor)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORIVECT</td>
<td>Large-radius circular interpol. (identical to ORIPLANE)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.1 List of instructions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Type</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIVRT1</td>
<td>Orientation angles using virtual orientation axes (def. 1)</td>
<td>m 50</td>
<td></td>
</tr>
<tr>
<td>ORIVRT2</td>
<td>Orientation angles using virtual orientation axes (definition 1)</td>
<td>m 50</td>
<td></td>
</tr>
<tr>
<td>ORIMCS</td>
<td>Tool orientation in machine coordinate system</td>
<td>m 25</td>
<td></td>
</tr>
<tr>
<td>ORIWKS</td>
<td>Tool orientation in workpiece coordinate system</td>
<td>m 25</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Oscillation ON / OFF</td>
<td>Integer, without sign</td>
<td></td>
</tr>
<tr>
<td>OSC</td>
<td>Constant smoothing for tool orientation</td>
<td>m 34</td>
<td></td>
</tr>
<tr>
<td>OSCILL</td>
<td>Axis assignment for oscillation - activate oscillation</td>
<td>Axis: 1–3 infeed axes</td>
<td></td>
</tr>
<tr>
<td>OSCTRL</td>
<td>Oscillation control options</td>
<td>Integer, without sign</td>
<td></td>
</tr>
<tr>
<td>OSE</td>
<td>Oscillation: End point</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>OSNSC</td>
<td>Oscillation: Number of spark-out cycles number spark out cycles</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>OSOF</td>
<td>Constant smoothing for tool orientation OFF</td>
<td>m 34</td>
<td></td>
</tr>
<tr>
<td>OSP1</td>
<td>Oscillation: Left-hand reversal point (oscillating: position 1)</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>OSP2</td>
<td>Oscillation: Right-hand reversal point (oscillating: position 2)</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>OSS</td>
<td>Tool orientation smoothing at end of block</td>
<td>m 34</td>
<td></td>
</tr>
<tr>
<td>OSSE</td>
<td>Tool orientation smoothing at beginning and end of block</td>
<td>m 34</td>
<td></td>
</tr>
<tr>
<td>OST1</td>
<td>Oscillation: Stop in left-hand reversal point</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>OST2</td>
<td>Oscillation: Stop in right-hand rev. point</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>OVR</td>
<td>Spindle override</td>
<td>1, ..., 200%</td>
<td></td>
</tr>
<tr>
<td>OVRA</td>
<td>Axial spindle override</td>
<td>1, ..., 200%</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>Number of subprogram passes</td>
<td>1 ... 9999, integer</td>
<td>E.g. L781 P... ; separate block</td>
</tr>
<tr>
<td>PCALL</td>
<td>PCALL calls subprograms with the absolute path and parameter transfer</td>
<td>No absolute path response like CALL</td>
<td></td>
</tr>
<tr>
<td>PDELAYOF</td>
<td>Delay for punching OFF (punch with delay OFF)</td>
<td>m 36</td>
<td></td>
</tr>
<tr>
<td>PDELAYON</td>
<td>Delay for punching ON (punch with delay ON)</td>
<td>m 36</td>
<td></td>
</tr>
<tr>
<td>PL</td>
<td>Parameter interval length</td>
<td>Real, without sign</td>
<td>s</td>
</tr>
<tr>
<td>PM</td>
<td>per minute</td>
<td>Feed per minute</td>
<td></td>
</tr>
<tr>
<td>PO</td>
<td>Polynomial</td>
<td>Real, without sign</td>
<td>s</td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
<td>--------------------</td>
<td>---</td>
</tr>
<tr>
<td>POLF</td>
<td>Position LIFTFAST</td>
<td>Real, without sign</td>
<td>m</td>
</tr>
<tr>
<td>POLY</td>
<td>Polynomial interpolation</td>
<td>m</td>
<td>1</td>
</tr>
<tr>
<td>POLYPATH</td>
<td>Polynomial interpolation can be selected for the AXIS or VECT axis groups</td>
<td>POLYPATH (&quot;AXES&quot;)</td>
<td>POLYPATH (&quot;VECT&quot;)</td>
</tr>
<tr>
<td>PON</td>
<td>Punching ON (punch ON)</td>
<td>m</td>
<td>35</td>
</tr>
<tr>
<td>PONS</td>
<td>Punching ON in IPO cycle (punch ON slow)</td>
<td>m</td>
<td>35</td>
</tr>
<tr>
<td>POS</td>
<td>Position axis</td>
<td>POS[X]=20</td>
<td></td>
</tr>
<tr>
<td>POSA</td>
<td>Position axis across block boundaries</td>
<td>POSA[Y]=20</td>
<td></td>
</tr>
<tr>
<td>POSP</td>
<td>Positioning in part sections (oscillation) (Position axis in parts)</td>
<td>Real: End position, part length; Integer; option</td>
<td></td>
</tr>
<tr>
<td>POT</td>
<td>Square (arithmetic function)</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>PR</td>
<td>Per revolution</td>
<td>Revolutionary feedrate</td>
<td></td>
</tr>
<tr>
<td>PRESETON</td>
<td>Set actual value for programmed axes</td>
<td>An axis name is programmed with the corresponding value in the next parameter. Up to 8 axes possible</td>
<td>PRESETON(X,10,Y,4.5)</td>
</tr>
<tr>
<td>PRIIO</td>
<td>Vocabulary word for setting the priority for interrupt processing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC</td>
<td>First instruction in a program</td>
<td>Block number - PROC - identifier</td>
<td></td>
</tr>
<tr>
<td>PTP</td>
<td>Point to point movement</td>
<td>m</td>
<td>49</td>
</tr>
<tr>
<td>PUTFTOC</td>
<td>Tool offset fine for parallel dressing (continuous dressing)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUTFTOCF</td>
<td>Put fine tool correction function dependent: Fine tool offset dependent on a function for continuous dressing defined with FCIDEF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PW</td>
<td>Point weight</td>
<td>Real, without sign</td>
<td>s</td>
</tr>
<tr>
<td>QECLRNOF</td>
<td>Quadrant error compensation learning OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QECLRON</td>
<td>Quadrant error compensation learning ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QU</td>
<td>Fast additional (auxiliary) function output</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.1 List of instructions

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R...</td>
<td>Arithmetic parameters</td>
<td>R parameter number can be set via MD</td>
</tr>
<tr>
<td></td>
<td>SW 5 and higher:</td>
<td>±0.0000001, ..., 9999 9999</td>
</tr>
<tr>
<td></td>
<td>also as settable address identifier and with numerical extension</td>
<td></td>
</tr>
<tr>
<td>RDISABLE</td>
<td>Read-in disable</td>
<td>Alarms are searched according to ascending numbers</td>
</tr>
<tr>
<td>READAL</td>
<td>Read alarm</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>Data type: floating point variable with leading sign (real numbers)</td>
<td>Corresponds to the 64-bit floating point format of the processor</td>
</tr>
<tr>
<td>REDEF</td>
<td>Setting for machine data, which user groups they are displayed for</td>
<td></td>
</tr>
<tr>
<td>RELEASE</td>
<td>Release machine axes</td>
<td>Multiple axes can be programmed</td>
</tr>
<tr>
<td>REP</td>
<td>Vocabulary word for initialization of all elements of an array with the same value</td>
<td></td>
</tr>
<tr>
<td>REPEAT</td>
<td>Repeat a program loop</td>
<td>until (UNTIL) a condition is fulfilled</td>
</tr>
<tr>
<td>REPEATB</td>
<td>Repeat a program line</td>
<td>nnn times</td>
</tr>
<tr>
<td>REPOSA</td>
<td>Reposition all axes linearly</td>
<td>s 2</td>
</tr>
<tr>
<td>REPOSH</td>
<td>Reposition along semi-circle</td>
<td>s 2</td>
</tr>
<tr>
<td>REPOSQA</td>
<td>Reposition all axes along quadrant</td>
<td>Reposition all axes along quadrant, geometry axes along quadrant</td>
</tr>
<tr>
<td>REPOSQ</td>
<td>Reposition along quadrant</td>
<td>s 2</td>
</tr>
<tr>
<td>REPOSL</td>
<td>Reposition linearly</td>
<td>s 2</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset technology cycle</td>
<td>One or several IDs can be programmed</td>
</tr>
<tr>
<td>RET</td>
<td>End of subprogram</td>
<td>Use in place of M17 – without function output to PLC</td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved

SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Parameters</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RINDEX</td>
<td>Define index of character in input string</td>
<td>0, ..., INT</td>
<td>String: Parameter 1, character: Parameter 2</td>
</tr>
<tr>
<td>RMB</td>
<td>Reposition at beginning of block</td>
<td>m</td>
<td>26</td>
</tr>
<tr>
<td>RMB</td>
<td>(Repos mode begin of block)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RME</td>
<td>Reposition at end of block</td>
<td>m</td>
<td>26</td>
</tr>
<tr>
<td>RME</td>
<td>(Repos mode end of block)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMI</td>
<td>Reposition at interruption point</td>
<td>m</td>
<td>26</td>
</tr>
<tr>
<td>RMI</td>
<td>(Repos mode interrupt)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMN</td>
<td>Reapproach to nearest path point</td>
<td>m</td>
<td>26</td>
</tr>
<tr>
<td>RMN</td>
<td>(Repos mode end of nearest orbital block)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RND</td>
<td>Round contour corner</td>
<td>Real, without sign</td>
<td>RND=...</td>
</tr>
<tr>
<td>RNDM</td>
<td>Modal rounding</td>
<td>Real, without sign</td>
<td>RNDM=...</td>
</tr>
<tr>
<td>RNDM</td>
<td></td>
<td></td>
<td>RNDM=0: disable modal rounding</td>
</tr>
<tr>
<td>ROT</td>
<td>Programmable rotation</td>
<td>Rotation around 1st geom. axis: -180° .. 180°</td>
<td>ROT X... Y... Z...</td>
</tr>
<tr>
<td>ROT</td>
<td></td>
<td>2nd G axis: -89.999°, ..., 90°</td>
<td>ROT RPL= ; separate block</td>
</tr>
<tr>
<td>ROT</td>
<td></td>
<td>3rd G axis: -180° .. 180°</td>
<td></td>
</tr>
<tr>
<td>ROTS</td>
<td>Programmable frame rotations with solid angles (rotation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROUND</td>
<td>Round decimal places</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>RP</td>
<td>Polar radius (radius polar)</td>
<td>Real</td>
<td>m,s³</td>
</tr>
<tr>
<td>RPL</td>
<td>Rotation in plane (rotation plane)</td>
<td>Real, without sign</td>
<td></td>
</tr>
<tr>
<td>RT</td>
<td>Parameter for access to frame data: Rotation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>Spindle speed or (with G4, G96) another meaning</td>
<td>0.1 ... 99999999.9</td>
<td>Spindle speed in rev/min</td>
</tr>
<tr>
<td>s</td>
<td></td>
<td>G4: Dwell time in spindle rotations</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td>G96: Cutting rate in m/min</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td>S...: Spindle speed for master spindle</td>
<td>m,s</td>
</tr>
<tr>
<td>s</td>
<td></td>
<td>S1...: Spindle speed for spindle 1</td>
<td></td>
</tr>
<tr>
<td>SAVE</td>
<td>Attribute for saving information at subroutine calls</td>
<td>The following are saved: All modal G functions and the current frame</td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Syntax</td>
<td>Parameters</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>SBLOF</td>
<td>Suppress single block (single block OFF)</td>
<td>The following blocks are executed in single block like a block.</td>
<td></td>
</tr>
<tr>
<td>SBLON</td>
<td>Clear single block suppression (single block ON)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td>Parameter for access to frame data: Scaling (scale)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCALE</td>
<td>Programmable scaling (scale)</td>
<td>SCALE X... Y... Z... ; separate block</td>
<td>s 3</td>
</tr>
<tr>
<td>SD</td>
<td>Spline degree Integer, without sign</td>
<td></td>
<td>s</td>
</tr>
<tr>
<td>SEFORM</td>
<td>Structuring instruction in Step editor to generate the step view for HMI Advanced</td>
<td>Evaluated in Step editor. SEFORM(&lt;section_name&gt;, &lt;level&gt;, &lt;icon&gt; )</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td>Vocabulary word for initialization of all elements of an array with listed values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETAL</td>
<td>Set alarm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETDNO</td>
<td>Set D number of tool (T) and its cutting edge to new</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETINT</td>
<td>Define which interrupt routine is to be activated when an NCK input is present</td>
<td>Edge 0 → 1 is analyzed</td>
<td></td>
</tr>
<tr>
<td>SETM</td>
<td>Set one/several markers for channel coordination</td>
<td>Machining in the local channel is not influenced by this.</td>
<td></td>
</tr>
<tr>
<td>SETMS</td>
<td>Switch back to master spindle programmed in machine data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETMS(n)</td>
<td>Spindle n must act as master spindle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETPIECE</td>
<td>Set piece number for all tools assigned to the spindle.</td>
<td>Without spindle number: Valid for master spindle</td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>Start point offset for thread cutting (spline offset)</td>
<td>0.0000, ..., 359.999°</td>
<td>m</td>
</tr>
<tr>
<td>SIN</td>
<td>Sine (trigon. function)</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>SOFT</td>
<td>Soft axis acceleration</td>
<td></td>
<td>m 21</td>
</tr>
<tr>
<td>SOFTA</td>
<td>Switch on soft axis acceleration for the programmed axes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SON ⁶</td>
<td>Nibbling ON (stroke ON)</td>
<td></td>
<td>m 35</td>
</tr>
<tr>
<td>SONS ⁶</td>
<td>Nibbling ON in IPO cycle (stroke ON slow)</td>
<td></td>
<td>m 35</td>
</tr>
<tr>
<td>SPATH ¹</td>
<td>Path reference for FGROUP axes is length of arc</td>
<td></td>
<td>m 45</td>
</tr>
<tr>
<td>SPCOF</td>
<td>Switch master spindle or spindle (n) from speed control over to position control</td>
<td></td>
<td>SPCON SPCON (n)</td>
</tr>
<tr>
<td>SPCON</td>
<td>Switch master spindle or spindle (n) from position control over to speed control</td>
<td></td>
<td>SPCON SPCON (n)</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Format</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>SPIF1</td>
<td>High-speed NCK inputs/outputs for punching/nibbling</td>
<td>m</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>byte 1 (stroke/punch interface 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>see /FB/, N4: Punching and Nibbling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPIF2</td>
<td>High-speed NCK inputs/outputs for punching/nibbling</td>
<td>m</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>byte 2 (stroke/punch interface 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>see /FB/, N4: Punching and Nibbling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPLINE-PATH</td>
<td>Define spline grouping</td>
<td>Max. of 8 axes</td>
<td></td>
</tr>
<tr>
<td>SPOE</td>
<td>Stroke OFF, punching, nibbling OFF</td>
<td>m</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>(stroke/punch OFF)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPN</td>
<td>Number of path sections per block</td>
<td>Integer</td>
<td>s</td>
</tr>
<tr>
<td></td>
<td>(stroke/punch number)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPP</td>
<td>Length of a path section</td>
<td>Integer</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>(stroke/punch path)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPOS</td>
<td>Spindle position</td>
<td>SPOS=10 or SPOS[n]=10</td>
<td>m</td>
</tr>
<tr>
<td>SPOSA</td>
<td>Spindle position across block boundaries</td>
<td>SPOSA=5 or SPOSA[n]=5</td>
<td>m</td>
</tr>
<tr>
<td>SQR</td>
<td>Square root; arithmetic function</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>SR</td>
<td>Sparking-out retraction path for</td>
<td>Real,</td>
<td>s</td>
</tr>
<tr>
<td></td>
<td>synchronized action</td>
<td>without sign</td>
<td></td>
</tr>
<tr>
<td>SRA</td>
<td>Sparking-out retraction path with input</td>
<td>SRA[Y]=0.2</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>axial for synchronized action</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>Sparking-out time for synchronized action</td>
<td>Real,</td>
<td>s</td>
</tr>
<tr>
<td></td>
<td>without sign</td>
<td>without sign</td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>Sparking out time axial for synchronized action</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>Start selected programs simultaneously in several</td>
<td>ineffective for the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>channels from current program</td>
<td>local channel</td>
<td></td>
</tr>
<tr>
<td>STAT</td>
<td>Position of articulated joints</td>
<td>Integer</td>
<td>s</td>
</tr>
<tr>
<td>STARTFIFO</td>
<td>Execute; fill preprocessing buffer in parallel</td>
<td>m</td>
<td>4</td>
</tr>
<tr>
<td>STOPFIFO</td>
<td>Stop processing; fill preprocessing buffer until</td>
<td>m</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>STARTFIFO, preprocessing buffer &quot;full&quot; or end of program is detected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STPRE</td>
<td>Stop preprocessing until all prepared blocks are</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>executed in main run.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STPREOF</td>
<td>Stop preprocessing OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>Data type: String</td>
<td>Max. 200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>characters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRLEN</td>
<td>Define string length</td>
<td>INT</td>
<td></td>
</tr>
<tr>
<td>SUBSTR</td>
<td>Define index of character in input string</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td></td>
<td>String: Parameter 1, character: Parameter 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Data Type</td>
<td>Parameters</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>------------</td>
</tr>
<tr>
<td>SUPA</td>
<td>Suppression of current zero offset incl. programm. offsets, handwheel offsets (DRF), external zero offsets and PRESET offset</td>
<td></td>
<td>s 9</td>
</tr>
<tr>
<td>SYNCFCT</td>
<td>Evaluation of a polynomial as a function of a condition in the motion-synchronous action</td>
<td>VAR REAL</td>
<td></td>
</tr>
<tr>
<td>SYNR</td>
<td>The variable is read synchronously, i.e. at execution time (synchronous read)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYNRW</td>
<td>The variable is read and written synchronously, i.e. at execution time (synchronous read-write)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYNW</td>
<td>The variable is written synchronously, i.e. at execution time (synchronous write)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Call tool (change only if so defined in machine data; otherwise M6 command required)</td>
<td>1 ... 32 000</td>
<td>Call via T No.: or via tool name:</td>
</tr>
<tr>
<td>TAN</td>
<td>Tangent (trigon. function)</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>TANG</td>
<td>Determine tangent for the follow-up from both specified leading axes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TANGOF</td>
<td>Tangent follow-up mode OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TANGON</td>
<td>Tangent follow-up mode ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCARR</td>
<td>Request toolholder (number &quot;m&quot;)</td>
<td>Integer</td>
<td>m=0: Deselect active toolholder TCARR=1</td>
</tr>
<tr>
<td>TCOABS ¹</td>
<td>Determine tool length components from current tool orientation</td>
<td></td>
<td>Required after resetting machine, e.g.</td>
</tr>
<tr>
<td>TCOFR</td>
<td>Determine tool length components from orientation of active frame</td>
<td></td>
<td>by manual setting</td>
</tr>
<tr>
<td>TCOFRX</td>
<td>Determine tool orientation of an active frame during tool selection, tool points in X direction</td>
<td>Tool perpendicular to sloping surface</td>
<td>m 42</td>
</tr>
<tr>
<td>TCOFRY</td>
<td>Determine tool orientation of an active frame during tool selection, tool points in Y direction</td>
<td>Tool perpendicular to sloping surface</td>
<td>m 42</td>
</tr>
<tr>
<td>TCOFRZ</td>
<td>Determine tool orientation of an active frame during tool selection, tool points in Z direction</td>
<td>Tool perpendicular to sloping surface</td>
<td>m 42</td>
</tr>
<tr>
<td>TILT ⁵</td>
<td>Side angle</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>TMOF</td>
<td>Deselect tool monitoring function</td>
<td>T number required only if tool with this number is not active</td>
<td>TMOF (T No.)</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Details</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>TMON</td>
<td>Select tool monitoring function</td>
<td>T No. = 0: Deactivate monitoring function for all tools</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMON (T No.)</td>
<td></td>
</tr>
<tr>
<td>TO</td>
<td>Defines the end value in a FOR counter loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOOFF</td>
<td>Deactivate on-line tool offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOFFON</td>
<td>Activate online tool length compensation (Tool Offset ON)</td>
<td>3-dimensional offset direction</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOFFON (Z, 25) with offset direction Z offset value 25</td>
<td></td>
</tr>
<tr>
<td>TOFRAME</td>
<td>Set current programmable frame to tool coordinate system</td>
<td>Frame rotations in tool direction</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFRAMEX</td>
<td>X axis parallel to tool direction, secondary axis Y, Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFRAMEY</td>
<td>Y axis parallel to tool direction, secondary axis Z, X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFRAMEZ</td>
<td>Z axis parallel to tool direction, secondary axis X, Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFROF</td>
<td>Frame rotations in tool direction OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFROT</td>
<td>Z axis parallel to tool orientation</td>
<td>Frame rotations ON</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rotation component of programmed frame</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFROTX</td>
<td>X axis parallel to tool orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFROTY</td>
<td>Y axis parallel to tool orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOFROTZ</td>
<td>Z axis parallel to tool orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 53</td>
<td></td>
</tr>
<tr>
<td>TOLOWER</td>
<td>Convert letters of the string into lowercase</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOWSTD</td>
<td>Initial setting value for corrections in tool length</td>
<td>Including tool wear</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 56</td>
<td></td>
</tr>
<tr>
<td>TOWBCS</td>
<td>Wear values in basic coordinate system BCS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 56</td>
<td></td>
</tr>
<tr>
<td>TOWKCS</td>
<td>Wear values in the coordinate system of the tool head for kinetic transformation (differs from MCS by tool rotation)</td>
<td>m 56</td>
<td></td>
</tr>
<tr>
<td>TOWMCS</td>
<td>Wear values in machine coordinate system (MCS).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 56</td>
<td></td>
</tr>
<tr>
<td>TOWTCS</td>
<td>Wear values in the tool coordinate system (tool carrier ref. point T at the tool holder)</td>
<td>m 56</td>
<td></td>
</tr>
<tr>
<td>TOWWCS</td>
<td>Wear values in workpiece coordinate system WCS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m 56</td>
<td></td>
</tr>
<tr>
<td>TOUPPER</td>
<td>Convert letters of the string into uppercase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR</td>
<td>Parameter for access to frame data: Translation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRAANG</td>
<td>Transformation inclined axis</td>
<td>Several transformations settable per channel</td>
<td></td>
</tr>
<tr>
<td>TRACEOF</td>
<td>Circularity test: Transfer of values OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACEON</td>
<td>Circularity test: Transfer of values ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACON</td>
<td>Transformation concatenated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACYL</td>
<td>Cylinder: Peripheral surface transformation</td>
<td>see TRAANG</td>
<td></td>
</tr>
</tbody>
</table>
### 15.1 List of instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAFOOF</td>
<td>Switch off transformation</td>
<td>TRANS X, Y, Z; separate block</td>
</tr>
<tr>
<td>TRAILOF</td>
<td>Synchronous coupled motion of axes OFF (trailing OFF)</td>
<td></td>
</tr>
<tr>
<td>TRAILON</td>
<td>Synchronous coupled motion of axes ON (trailing ON)</td>
<td></td>
</tr>
<tr>
<td>TRANS</td>
<td>Programmable offset (translation)</td>
<td></td>
</tr>
<tr>
<td>TRANSMIT</td>
<td>Polar transformation</td>
<td>see TRAANG</td>
</tr>
<tr>
<td>TRAORI</td>
<td>4-axis, 5-axis transformation (transformation oriented)</td>
<td>see TRAANG</td>
</tr>
<tr>
<td>TRUE</td>
<td>Logical constant: True</td>
<td>BOOL Can be replaced with integer constant 1</td>
</tr>
<tr>
<td>TRUNC</td>
<td>Truncate decimal places</td>
<td>Real</td>
</tr>
<tr>
<td>TU</td>
<td>Axis angle</td>
<td>Integer TU=2</td>
</tr>
<tr>
<td>TURN</td>
<td>No. of turns for helix</td>
<td>0, ..., 999 s</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>Enable synchronized action with ID (continue technology cycle)</td>
<td></td>
</tr>
<tr>
<td>UNTIL</td>
<td>Condition for end of REPEAT loop</td>
<td></td>
</tr>
<tr>
<td>UPATH</td>
<td>Curve parameter is path reference for FGROUP axes</td>
<td>m 45</td>
</tr>
<tr>
<td>VAR</td>
<td>Vocabulary word: Type of parameter passing</td>
<td>With VAR: Call by reference</td>
</tr>
<tr>
<td>WAITC</td>
<td>Wait until coupling block change criterion for axes / spindles is fulfilled (wait for couple condition)</td>
<td>Up to 2 axes/spindles can be programmed. WAITC(1,1,2)</td>
</tr>
<tr>
<td>WAITM</td>
<td>Wait for marker in specified channel; terminate previous block with exact stop.</td>
<td>WAITM(1,1,2)</td>
</tr>
<tr>
<td>WAITMC</td>
<td>Wait for marker in specified channel; exact stop only if other channels have not yet reached the marker</td>
<td>WAITMC(1,1,2)</td>
</tr>
<tr>
<td>WAITP</td>
<td>Wait for end of travel</td>
<td>WAITP(X); separate block</td>
</tr>
<tr>
<td>WAITS</td>
<td>Wait until spindle position is reached</td>
<td>WAITS (main spindle) WAITS (n,n,n)</td>
</tr>
<tr>
<td>WALIMOF</td>
<td>Working area limitation OFF</td>
<td>; separate block m 28</td>
</tr>
<tr>
<td>WALIMON</td>
<td>Working area limitation ON</td>
<td>; separate block m 28</td>
</tr>
<tr>
<td>WHILE</td>
<td>Start of WHILE program loop</td>
<td>End: ENDWHILE</td>
</tr>
<tr>
<td>WRITE</td>
<td>Write block in file system</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Axis</td>
<td>Real m,s³</td>
</tr>
<tr>
<td>XOR</td>
<td>Logical exclusive OR</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>Axis</td>
<td>Real m,s³</td>
</tr>
<tr>
<td>Z</td>
<td>Axis</td>
<td>Real m,s³</td>
</tr>
</tbody>
</table>
**Legend:**

1. Default setting at start of program (in delivery state of control system provided that another setting is not programmed).
2. The group numbering corresponds to the numbering in table "Overview of instructions" in Section 11.3
3. Absolute end points: Modal; incremental end points: Non-modal; otherwise modal/non-modal depending on syntax of G function
4. IPO parameters act incrementally as arc centers. They can be programmed in absolute mode with AC. When they have other meanings (e.g. pitch), the address modification is ignored.
5. Vocabulary word does not apply to SINUMERIK FM-NC/810D
6. Vocabulary word does not apply to SINUMERIK FM-NC/810D/NCU571
7. Vocabulary word does not apply to SINUMERIK 810D
8. The OEM user can incorporate two extra interpolation types and modify their names.
9. Vocabulary word applies only to SINUMERIK FM-NC
10. The extended address block format may not be used for these functions.
### 15.2 List of system variables

#### Legend:

<table>
<thead>
<tr>
<th>Parts pr.</th>
<th>Parts program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syn</td>
<td>Synchronized action</td>
</tr>
<tr>
<td>O</td>
<td>The index can be calculated online in synchronized actions. (+)</td>
</tr>
<tr>
<td>S</td>
<td>Software version</td>
</tr>
<tr>
<td>R</td>
<td>Read access possible</td>
</tr>
<tr>
<td>W</td>
<td>Write access possible</td>
</tr>
<tr>
<td>RS</td>
<td>A preprocessor stop takes place implicitly on read access</td>
</tr>
<tr>
<td>WS</td>
<td>A preprocessor stop takes place implicitly on write access</td>
</tr>
<tr>
<td>+</td>
<td>In column O: The index can be calculated online in synchronized actions.</td>
</tr>
</tbody>
</table>

#### 15.2.1 R parameters

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R</td>
<td>REAL</td>
<td>Rn or R[n]</td>
<td>R</td>
<td>W</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of R parameters is defined in machine data</td>
<td>R</td>
<td>W</td>
<td>R</td>
<td>4</td>
</tr>
</tbody>
</table>

#### 15.2.2 Channel-specific synchronized action variables

<table>
<thead>
<tr>
<th>$AC_PARAM</th>
<th>REAL</th>
<th>$AC_PARAM[n]</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Arithmetic variable for motion-synchronized actions</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The dimension is fixed by the machine data $MC_MM_NUM_AC_PARAM.</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_SYSTEM_PARAM</td>
<td>REAL</td>
<td>$AC_SYSTEM_PARAM[n]</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Arithmetic variable for motion-synchronized actions</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved for SIEMENS applications</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The dimension is fixed by the machine data $MC_MM_NUM_AC_SYSTEM_PARAM.</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_MARKER</td>
<td>INT</td>
<td>$AC_MARKER[n]</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Marker variable for motion-synchronized actions</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The dimension is fixed by the machine data $MC_MM_NUM_AC_MARKER.</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 15.2 List of system variables

### 15.2.3 Frames 1

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_SYSTEM_MARKER</td>
<td>INT $AC_SYSTEM_MARKER[n]</td>
<td>Marker variable for motion-synchronized actions Reserved for SIEMENS applications The dimension is fixed by the machine data $MC_MM_NUM_AC_SYSTEM_MARKER.</td>
<td>R</td>
<td>S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### $P_UIFR
- **FRAME** $P_UIFR[n]
- Settable frames, can be activated via G500, G54 .. G599.
- 5 to 100 settable frames with MD $MC_MM_NUM_USER_FRAMES

#### $P_CHBFR
- **FRAME** $P_CHBFR[n]
- Channel base frames, can be activated via G500, G54 .. G599.
- 0 to 8 channel base frames via MD $MC_MM_NUM_BASE_FRAMES.

#### $P_SETFR
- **FRAME** $P_SETFR
- Axes: (geometry axis, channel axis, machine axis)

#### $P_EXTFR
- **FRAME** $P_EXTFR
- Axes: (geometry axis, channel axis, machine axis)

#### $P_PARTFR
- **FRAME** $P_PARTFR
- Axes: (geometry axis, channel axis, machine axis)

#### $P_TOOLFR
- **FRAME** $P_TOOLFR
- Axes: (geometry axis, channel axis, machine axis)

#### $P_WPFR
- **FRAME** $P_WPFR
- Axes: (geometry axis, channel axis, machine axis)

#### $P_CYCFR
- **FRAME** $P_CYCFR
- Axes: (geometry axis, channel axis, machine axis)
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{TRAFR}}$</td>
<td>FRAME</td>
<td>$P_{\text{TRAFR}}$</td>
<td>R W 6</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: (geometry axis, channel axis, machine axis)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_{\text{NCBFR}}$</td>
<td>FRAME</td>
<td>$P_{\text{NCBFR}}[n]$ NCU base frames, can be activated via G500, G54 .. G599. 0 to 8 NCU base frames via MD $\text{MIN_MM_NUM_GLOBAL_BASE_FRAMES}$.</td>
<td>R W 5</td>
<td>O</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 15.2.4 Toolholder data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_CARR1}$</td>
<td>REAL</td>
<td>$\text{TC_CARR1}[n]$ x component of offset vector l1 Notice! All system parameters with the '$\text{TC_}$' prefix are contained in the TOA area. The special property of this area is that it is possible, conditional on machine data 28085 = MM_LINK_TOA_UNIT, for various NCK channels to access these parameters. If an NCK parameterization of this type has been selected, then it must be clearly understood that when this data is changed, the changes can also have an adverse effect on another channel; or there must be evidence that the change only has a local effect on the channel that is changed. The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W 4</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_CARR2}$</td>
<td>REAL</td>
<td>$\text{TC_CARR2}[n]$ y component of offset vector l1 The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W 4</td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_CARR3</td>
<td>REAL</td>
<td>$TC_CARR3[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>z component of offset vector l1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR4</td>
<td>REAL</td>
<td>$TC_CARR4[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x component of offset vector l2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR5</td>
<td>REAL</td>
<td>$TC_CARR5[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>y component of offset vector l2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR6</td>
<td>REAL</td>
<td>$TC_CARR6[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>z component of offset vector l2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR7</td>
<td>REAL</td>
<td>$TC_CARR7[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x component of axis of rotation v1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR8</td>
<td>REAL</td>
<td>$TC_CARR8[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>y component of axis of rotation v1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR9</td>
<td>REAL</td>
<td>$TC_CARR9[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>z component of axis of rotation v1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR10</td>
<td>REAL</td>
<td>$TC_CARR10[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x component of axis of rotation v2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR11</td>
<td>REAL</td>
<td>$TC_CARR11[n]</td>
<td>R</td>
<td>W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>y component of axis of rotation v2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The maximum number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_CARR12$</td>
<td>REAL</td>
<td>$TC_CARR12[n]$ z component of axis of rotation v2</td>
<td></td>
<td>R W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR13$</td>
<td>REAL</td>
<td>$TC_CARR13[n]$ Angle of rotation alpha1 (in degrees)</td>
<td></td>
<td>R W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR14$</td>
<td>REAL</td>
<td>$TC_CARR14[n]$ Angle of rotation alpha2 (in degrees)</td>
<td></td>
<td>R W</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR15$</td>
<td>REAL</td>
<td>$TC_CARR15[n]$ x component of offset vector l3</td>
<td></td>
<td>R W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR16$</td>
<td>REAL</td>
<td>$TC_CARR16[n]$ y component of offset vector l3</td>
<td></td>
<td>R W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR17$</td>
<td>REAL</td>
<td>$TC_CARR17[n]$ z component of offset vector l3</td>
<td></td>
<td>R W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR18$</td>
<td>REAL</td>
<td>$TC_CARR18[n]$ x component of offset vector l4</td>
<td></td>
<td>R W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR19$</td>
<td>REAL</td>
<td>$TC_CARR19[n]$ y component of offset vector l4</td>
<td></td>
<td>R W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_CARR20$</td>
<td>REAL</td>
<td>$TC_CARR20[n]$ z component of offset vector l4</td>
<td></td>
<td>R W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_CARR21</td>
<td>AXIS</td>
<td>$TC_CARR21[n]$ Axis identifier for the 1st axis of rotation&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$TC_CARR22</td>
<td>AXIS</td>
<td>$TC_CARR22[n]$ Axis identifier for the 2nd axis of rotation&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$TC_CARR23</td>
<td>CHAR</td>
<td>$TC_CARR23[n]$ Kinematics type: P: rotatable workpiece (Part)&lt;br&gt;M: rotatable tool and rotatable part (Mixed)&lt;br&gt;T or any other character apart from P and M: rotatable tool&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = T; i.e. toolholder with orientable tool.</td>
<td>R W</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$TC_CARR24</td>
<td>REAL</td>
<td>$TC_CARR24[n]$ Offset of the 1st rotary axis in degrees&lt;br&gt;Indicates the angle in degrees of the 1st rotary axis at which the axis takes up its initial setting.&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$TC_CARR25</td>
<td>REAL</td>
<td>$TC_CARR25[n]$ Offset of the 2nd rotary axis in degrees&lt;br&gt;Indicates the angle in degrees of the 2nd rotary axis, at which the axis takes up its initial setting.&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$TC_CARR26</td>
<td>REAL</td>
<td>$TC_CARR26[n]$ Indicates the offset of the 1st rotary axis if its position cannot be changed continuously (Hirth tooth system).&lt;br&gt;It will only be analyzed if $TC_CARR28 does not equal zero.&lt;br&gt;For detailed explanation see the description of $TC_CARR28&lt;br&gt;The maximum number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_CARR27$</td>
<td>REAL</td>
<td>$TC_CARR27[n]$ Indicates the offset of the 1st rotary axis if its position cannot be changed continuously (Hirth tooth system). It will only be analyzed if $TC_CARR29$ does not equal zero. For detailed explanation see the description of $TC_CARR29$ The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td>RW</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| $TC\_CARR28$ | REAL  | $TC\_CARR28[n]$ Specifies the size of the minimum increment step (in degrees), by which the first rotary axis can be changed (e.g. for Hirth tooth systems). A programmed or calculated angle is rounded to the nearest value that arises with integer n from \[
\phi = s + n \cdot d
\] while 
\[
s = $TC\_CARR28$
\]
\[
d = $TC\_CARR26$
\] If $TC\_CARR28$ equals zero, $TC\_CARR26$ and $TC\_CARR28$ are not used. Instead, machine data $MC\_TOCARR\_ROT\_ANGLE\_INCR[i]$ and $MC\_TOCARR\_ROT\_ANGLE\_OFFSET[i]$ are accessed. The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data. | RW        |     |   |   |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\textsc{tc}_{\text{carr}29}$</td>
<td>REAL</td>
<td>$\textsc{tc}<em>{\text{carr}29}[n]$&lt;br&gt;Specifies the size of the minimum increment step (in degrees), by which the second rotary axis can be changed (e.g. for Hirth tooth systems).&lt;br&gt;A programmed or calculated angle is rounded to the nearest value that arises with integer n from&lt;br&gt;( \phi = s + n \cdot d )&lt;br&gt;while&lt;br&gt;s = $\textsc{tc}</em>{\text{carr}29}$&lt;br&gt;d = $\textsc{tc}<em>{\text{carr}27}$&lt;br&gt;If $\textsc{tc}</em>{\text{carr}29}$ equals zero, $\textsc{tc}<em>{\text{carr}28}$ and $\textsc{tc}</em>{\text{carr}29}$ are not used.&lt;br&gt;Instead, machine data&lt;br&gt;$\text{mc}<em>{\text{tocarr_rot_angle_incr}}[i]$ and&lt;br&gt;$\text{mc}</em>{\text{tocarr_rot_angle_offset}}[i]$&lt;br&gt;are accessed.&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$\textsc{tc}_{\text{carr}30}$</td>
<td>REAL</td>
<td>$\textsc{tc}<em>{\text{carr}30}[n]$&lt;br&gt;Indicates the minimum position of the 1st rotary axis. For a detailed description, see $\textsc{tc}</em>{\text{carr}32}$&lt;br&gt;The maximum number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$\textsc{tc}_{\text{carr}31}$</td>
<td>REAL</td>
<td>$\textsc{tc}<em>{\text{carr}31}[n]$&lt;br&gt;Indicates the minimum position of the 2nd rotary axis. For a detailed description, see $\textsc{tc}</em>{\text{carr}33}$&lt;br&gt;The max. number of toolholders can be set via machine data.&lt;br&gt;Default setting is = 0; i.e. NCK has no such data.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC}_\text{CARR32}$</td>
<td>REAL</td>
<td>$\text{TC}<em>\text{CARR32}[n]$ Indicates the maximum position of the 1st rotary axis. When calculating the angle of the 1st rotary axis of an orientable toolholder during alignment to a frame (TCOFR), only those solutions that fall within the range $\text{TC}</em>\text{CARR30}$ to $\text{TC}<em>\text{CARR32}$ are accepted as valid. The same applies for an angle of rotation programmed as absolute (TCOABS). If both $\text{TC}</em>\text{CARR30}$ and $\text{TC}_\text{CARR32}$ equal zero, the limitations will not be analyzed. The maximum number of toolholders can be set via machine data. Default setting is $= 0$; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6 .</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\text{TC}_\text{CARR33}$</td>
<td>REAL</td>
<td>$\text{TC}<em>\text{CARR33}[n]$ Indicates the maximum position of the 2nd rotary axis. When calculating the angle of the 2nd rotary axis of an orientable toolholder during alignment to a frame (TCOFR), only those solutions that fall within the range $\text{TC}</em>\text{CARR31}$ to $\text{TC}<em>\text{CARR33}$ are accepted as valid. The same applies for an angle of rotation programmed as absolute (TCOABS). If both $\text{TC}</em>\text{CARR31}$ and $\text{TC}_\text{CARR33}$ equal zero, the limitations will not be analyzed. The maximum number of toolholders can be set via machine data. Default setting is $= 0$; i.e. NCK has no such data.</td>
<td>R W</td>
<td>6 .</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_CARR34}$</td>
<td>STRING</td>
<td>$\text{TC_CARR34}[n]$ Contains a user-definable string. This is intended to be a free identifier for the orientable toolholder. However, within the NCK it is currently totally meaningless and is not evaluated either. The identifier should not be used for other purposes, as in a later expansion, the activation of an orientable toolholder should also be possible via names instead of numbers. The max. number of toolholders can be set via machine data. Default setting is $= 0$; i.e. NCK has no such data. 2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_CARR35}$</td>
<td>STRING</td>
<td>$\text{TC_CARR35}[n]$ Contains a user-definable string. This is intended to be a free identifier for the first rotary axis. However, within the NCK it is totally meaningless and is not evaluated either. It can therefore also be used for any other purposes. The max. number of toolholders can be set via machine data. Default setting is $= 0$; i.e. NCK has no such data. 2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_CARR36}$</td>
<td>STRING</td>
<td>$\text{TC_CARR36}[n]$ Contains a user-definable string. This is intended to be a free identifier for the second rotary axis. However, within the NCK it is totally meaningless and is not evaluated either. It can therefore also be used for any other purposes. The max. number of toolholders can be set via machine data. Default setting is $= 0$; i.e. NCK has no such data. 2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$STC_CARR37</td>
<td>INT</td>
<td>Contains an integer to identify the toolholder. However, within the NCK it is totally meaningless and is not evaluated either. The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$STC_CARR38</td>
<td>REAL</td>
<td>Contains a position (X component of the retraction position) However, within the NCK it is totally meaningless and is not evaluated either. The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$STC_CARR39</td>
<td>REAL</td>
<td>Contains a position (Y component of the retraction position) However, within the NCK it is totally meaningless and is not evaluated either. The max. number of toolholders can be set via machine data. Default setting is = 0; i.e. NCK has no such data.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

### 15.2.5 Channel-specific protection zones

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SSC_PA_ACTIV_IMMED</td>
<td>BOOL</td>
<td>Protection zone active immediately? TRUE: The protection zone is active immediately once the control is powered up and the axes are referenced FALSE: The protection zone is not active immediately n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SSC_PA_T_W</td>
<td>CHAR</td>
<td>Part/tool related protection zone 0: Part-related protection zone 3: Tool-related protection zone n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$SC_PA_ORI$</td>
<td>INT</td>
<td>$SC_PA_ORI[n]$ Orientation of protection zone 0: Polygon in plane from 1st and 2nd geo axis 1: Polygon in plane from 3rd and 1st geo axis 2: Polygon in plane from 2nd and 3rd geo axis n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SC_PA_LIM_3DIM$</td>
<td>INT</td>
<td>$SC_PA_LIM_3DIM[n]$ Code for restricting the protection zone in the axis that lies perpendicular to the polygon definition 0: = No limit 1: = Limit in positive direction 2: = Limit in negative direction 3: = Limit in both directions n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SC_PA_PLUS_LIM$</td>
<td>REAL</td>
<td>$SC_PA_PLUS_LIM[n]$ Positive limit of the protection zones in the axis that lies perpendicular to the polygon definition n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SC_PA_MINUS_LIM$</td>
<td>REAL</td>
<td>$SC_PA_MINUS_LIM[n]$ Negative limitation of protection zones in the negative direction in the axis that lies perpendicular to the polygon definition n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SC_PA_CONT_NUM$</td>
<td>INT</td>
<td>$SC_PA_CONT_NUM[n]$ Number of valid contour elements n: Number of protection zone 0 - (maximum value settable via MD)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SC_PA_CONT_TYP$</td>
<td>INT</td>
<td>$SC_PA_CONT_TYP [n,m]$ Contour element type (G1, G2, G3) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10 (MAXNUM_CONTOURNO_PROTECTAREA)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SC_PA_CONT_ORD$</td>
<td>REAL</td>
<td>$SC_PA_CONT_ORD[n,m]$ End point of contour element (ordinate) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10 (MAXNUM_CONTOURNO_PROTECTAREA)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $SC_PA_CONT_ABS       | REAL   | $SC_PA_CONT_ABS[n,m]
End point of contour element (abscissa)
  
  n: Number of protection zone 0 - (maximum value settable via MD)
  m: Number of contour element 0–10
  (MAXNUM_CONTOURNO_PROTECTAREA)                                           | R        | W    | 2 |
| $SC_PA_CENT_ORD       | REAL   | $SC_PA_CENT_ORD[n,m]
Center point of contour element (ordinate)
  
  n: Number of protection zone 0 - (maximum value settable via MD)
  m: Number of contour element 0–10
  (MAXNUM_CONTOURNO_PROTECTAREA)                                           | R        | W    | 2 |
| $SC_PA_CENT_ABS       | REAL   | $SC_PA_CENT_ABS[n,m]
Center point of contour element (abscissa)
  
  n: Number of protection zone 0 - (maximum value settable via MD)
  m: Number of contour element 0–10
  (MAXNUM_CONTOURNO_PROTECTAREA)                                           | R        | W    | 2 |

### 15.2.6 Tool parameters

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $TC_DP1      | INT    | $TC_DP1[t,d]
Tool type
With active 'Flat D number management' function, the syntax is as follows:
  $TC_DP1[d]
  t: Tool number 1–32000
  d: Cutting edge number/D number 1–32000                                        | R        | W    | 2 |
| $TC_DP2      | REAL   | $TC_DP2[t,d]
Tool edge position
With active 'Flat D number management' function, the syntax is as follows:
  $TC_DP2[d]
  t: Tool number 1–32000
  d: Cutting edge number/D number 1–32000                                        | R        | W    | 2 |
| $TC_DP3      | REAL   | $TC_DP3[t,d]
Geometry - Length 1
With active 'Flat D number management' function, the syntax is as follows:
  $TC_DP3[d]
  t: Tool number 1–32000
  d: Cutting edge number/D number 1–32000                                        | R        | W    | 2 |
### Table 5.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{T}_{\text{C}}\text{D}_4$</td>
<td>REAL</td>
<td>$\text{T}_{\text{C}}\text{D}<em>4[t,d]$ Geometry - Length 2 With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}</em>{\text{C}}\text{D}_4[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{T}_{\text{C}}\text{D}_5$</td>
<td>REAL</td>
<td>$\text{T}_{\text{C}}\text{D}<em>5[t,d]$ Geometry - Length 3 With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}</em>{\text{C}}\text{D}_5[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{T}_{\text{C}}\text{D}_6$</td>
<td>REAL</td>
<td>$\text{T}_{\text{C}}\text{D}<em>6[t,d]$ Geometry - Radius With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}</em>{\text{C}}\text{D}_6[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{T}_{\text{C}}\text{D}_7$</td>
<td>REAL</td>
<td>$\text{T}_{\text{C}}\text{D}<em>7[t,d]$ Slotting saw: Corner radius With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}</em>{\text{C}}\text{D}_7[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{T}_{\text{C}}\text{D}_8$</td>
<td>REAL</td>
<td>$\text{T}_{\text{C}}\text{D}<em>8[t,d]$ Slotting saw: Length With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}</em>{\text{C}}\text{D}_8[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{T}_{\text{C}}\text{D}_9$</td>
<td>REAL</td>
<td>$\text{T}_{\text{C}}\text{D}<em>9[t,d]$ Reserved With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}</em>{\text{C}}\text{D}_9[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{T}<em>{\text{C}}\text{D}</em>{10}$</td>
<td>REAL</td>
<td>$\text{T}<em>{\text{C}}\text{D}</em>{10}[t,d]$ Angle between face of tool and torus surface With active ‘Flat D number management’ function, the syntax is as follows: $\text{T}<em>{\text{C}}\text{D}</em>{10}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$\text{TC_DP11}$</td>
<td>REAL</td>
<td>$\text{TC_DP11}[t,d]$ $\text{TC_DP11}[t,d]$ Angle between tool longitudinal axis and upper end of torus surface With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DP11}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>RW</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_DP12}$</td>
<td>REAL</td>
<td>$\text{TC_DP12}[t,d]$ $\text{TC_DP12}[t,d]$ Wear - Length 1 - $\text{TC_DP3}$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DP12}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>RW</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_DP13}$</td>
<td>REAL</td>
<td>$\text{TC_DP13}[t,d]$ $\text{TC_DP13}[t,d]$ Wear - Length 2 - $\text{TC_DP4}$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DP13}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>RW</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_DP14}$</td>
<td>REAL</td>
<td>$\text{TC_DP14}[t,d]$ $\text{TC_DP14}[t,d]$ Wear - Length 3 - $\text{TC_DP5}$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DP14}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>RW</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_DP15}$</td>
<td>REAL</td>
<td>$\text{TC_DP15}[t,d]$ $\text{TC_DP15}[t,d]$ Wear - Radius - $\text{TC_DP6}$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DP15}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>RW</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_DP16}$</td>
<td>REAL</td>
<td>$\text{TC_DP16}[t,d]$ Slotting saw: Wear, corner radius - $\text{TC_DP7}$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DP16}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>RW</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_DP17</td>
<td>REAL</td>
<td>$TC_DP17[t,d] Slotting saw: Wear - Length - $TC_DP8</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active 'Flat D number management' function, the syntax is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>as follows: $TC_DP17[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP18</td>
<td>REAL</td>
<td>$TC_DP18[t,d] Wear - Reserved - $TC_DP9</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active 'Flat D number management' function, the syntax is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>as follows: $TC_DP18[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP19</td>
<td>REAL</td>
<td>$TC_DP19[t,d] Wear - Angle between face of tool and torus surface - $TC_DP10</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active 'Flat D number management' function, the syntax is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>as follows: $TC_DP19[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP20</td>
<td>REAL</td>
<td>$TC_DP20[t,d] Wear angle between tool longitudinal axis and upper end of</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>torus surface - $TC_DP11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active 'Flat D number management' function, the syntax is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>as follows: $TC_DP20[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP21</td>
<td>REAL</td>
<td>$TC_DP21[t,d] Base - Length 1</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active 'Flat D number management' function, the syntax is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>as follows: $TC_DP21[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_DP22</td>
<td>REAL</td>
<td>$TC_DP22[t,d] Base - Length 2</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active 'Flat D number management' function, the syntax is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>as follows: $TC_DP22[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $TC\_DP23$ | REAL | $TC\_DP23[t,d]$ Base - Length 3  
With active 'Flat D number management' function, the syntax is as follows:  
$t$: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 2 |   |
| $TC\_DP24$ | REAL | $TC\_DP24[t,d]$  
Clearance angle  
With active 'Flat D number management' function, the syntax is as follows:  
$t$: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 2 |   |
| $TC\_DP25$ | REAL | $TC\_DP25[t,d]$  
Reserved  
With active 'Flat D number management' function, the syntax is as follows:  
$t$: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 4 |   |
| $TC\_DPCE$ | INT | $TC\_DPCE[t,d]$ = 'Cutting edge number' of offset data block $t,d$  
With active 'Flat D number management' function, the syntax is as follows:  
$CE$ stands for $<C>utting<E>dge$  
Range of values of legal 'cutting edge numbers':  
1 to the value of machine data  
$MN\_MM\_MAX\_CUTTING\_EDGE\_PERTOOL$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |   |
| $TC\_DPH$ | INT | $TC\_DPH[t,d]$ = 'H cutting edge number' of offset data block $t,d$ for  
Fanuc0 M  
With active 'Flat D number management' function, the syntax is as follows:  
$t$: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 | 1 |
| $TC\_DPV$ | INT | $TC\_DPV[t,d]$ = tool cutting edge orientation  
With active 'Flat D number management' function, the syntax is as follows:  
$t$: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 6 | 1 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_DPV3}$</td>
<td>REAL</td>
<td>$\text{TC_DPV3}[t,d] = X$-component of tool cutting edge orientation With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPV3}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$\text{TC_DPV4}$</td>
<td>REAL</td>
<td>$\text{TC_DPV4}[t,d] = Y$-component of tool cutting edge orientation With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPV4}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$\text{TC_DPV5}$</td>
<td>REAL</td>
<td>$\text{TC_DPV5}[t,d] = Z$-component of tool cutting edge orientation With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPV5}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
</tbody>
</table>
### 15.2.7 Cutting edge data OEM user

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_DPC1}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $\text{TC_DPC1}[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPC1}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_DPC2}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $\text{TC_DPC2}[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPC2}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $\text{TC_DPC}[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPC}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $\text{TC_DPC10}[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $\text{TC_DPC10}[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>------------------------------------------------</td>
<td>----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_DPCS1$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $TC_DPCS1[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_DPCS1[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$TC_DPCS2$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $TC_DPCS2[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_DPCS2[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$TC_DPCSi$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $TC_DPCSi[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_DPCSi[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$TC_DPCS10$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is DOUBLE $TC_DPCS10[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_DPCS10[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_{SCP13}$</td>
<td>REAL</td>
<td>Offset for $TC_{DP3}$: $TC_{SCP13}[t,d]$ analogous to $TC_{DP12}[t,d]$ with active 'Flat D number management'</td>
<td>R/W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>function, the syntax is as follows: $TC_{SCP13}[d]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{SCP14}$</td>
<td>REAL</td>
<td>Offset for $TC_{DP4}$: $TC_{SCP14}[t,d]$ analogous to $TC_{DP13}[t,d]$ with active 'Flat D number management'</td>
<td>R/W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>function, the syntax is as follows: $TC_{SCP14}[d]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{SCP21}$</td>
<td>REAL</td>
<td>Offset for $TC_{DP11}$: $TC_{SCP21}[t,d]$ analogous to $TC_{DP20}[t,d]$ with active 'Flat D number management'</td>
<td>R/W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>function, the syntax is as follows: $TC_{SCP21}[d]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{SCP23}$</td>
<td>REAL</td>
<td>Offset for $TC_{DP3}$: $TC_{SCP23}[t,d]$ analogous to $TC_{DP12}[t,d]$ with active 'Flat D number management'</td>
<td>R/W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>function, the syntax is as follows: $TC_{SCP23}[d]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{SCP24}$</td>
<td>REAL</td>
<td>Offset for $TC_{DP4}$: $TC_{SCP24}[t,d]$ analogous to $TC_{DP13}[t,d]$ with active 'Flat D number management'</td>
<td>R/W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>function, the syntax is as follows: $TC_{SCP24}[d]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_{SCP31}$</td>
<td>REAL</td>
<td>Offset for $TC_{DP11}$: $TC_{SCP31}[t,d]$ analogous to $TC_{DP20}[t,d]$ with active 'Flat D number management'</td>
<td>R/W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>function, the syntax is as follows: $TC_{SCP31}[d]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Tables 11.02

#### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\text{TC}\_\text{SCP33}$ | REAL | Offset for $\text{TC}\_\text{DP3}$: $\text{TC}\_\text{SCP33}[t,d]$ analogous to $\text{TC}\_\text{DP12}[t,d]$  
With active 'Flat D number management' function, the syntax is as follows:  
$\text{TC}\_\text{SCP33}[d]$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |
| $\text{TC}\_\text{SCP34}$ | REAL | Offset for $\text{TC}\_\text{DP4}$: $\text{TC}\_\text{SCP34}[t,d]$ analogous to $\text{TC}\_\text{DP13}[t,d]$  
With active 'Flat D number management' function, the syntax is as follows:  
$\text{TC}\_\text{SCP34}[d]$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |
| ... | | | | | | |
| $\text{TC}\_\text{SCP41}$ | REAL | Offset for $\text{TC}\_\text{DP11}$: $\text{TC}\_\text{SCP41}[t,d]$ analogous to $\text{TC}\_\text{DP20}[t,d]$  
With active 'Flat D number management' function, the syntax is as follows:  
$\text{TC}\_\text{SCP41}[d]$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |
| $\text{TC}\_\text{SCP43}$ | REAL | Offset for $\text{TC}\_\text{DP3}$: $\text{TC}\_\text{SCP43}[t,d]$ analogous to $\text{TC}\_\text{DP12}[t,d]$  
With active 'Flat D number management' function, the syntax is as follows:  
$\text{TC}\_\text{SCP43}[d]$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |
| $\text{TC}\_\text{SCP44}$ | REAL | Offset for $\text{TC}\_\text{DP4}$: $\text{TC}\_\text{SCP44}[t,d]$ analogous to $\text{TC}\_\text{DP13}[t,d]$  
With active 'Flat D number management' function, the syntax is as follows:  
$\text{TC}\_\text{SCP44}[d]$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |
| ... | | | | | | |
| $\text{TC}\_\text{SCP51}$ | REAL | Offset for $\text{TC}\_\text{DP11}$: $\text{TC}\_\text{SCP51}[t,d]$ analogous to $\text{TC}\_\text{DP20}[t,d]$  
With active 'Flat D number management' function, the syntax is as follows:  
$\text{TC}\_\text{SCP51}[d]$  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R | W | 5 |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $TC\_SCP53 | REAL  | Offset for $TC\_DP3: $TC\_SCP53[t,d] analogous to $TC\_DP12[t,d]  
With active 'Flat D number management' function, the syntax is as follows:  
$TC\_SCP53[d]  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R W       |     |   | 5 |
| $TC\_SCP54 | REAL  | Offset for $TC\_DP4: $TC\_SCP54[t,d] analogous to $TC\_DP13[t,d]  
With active 'Flat D number management' function, the syntax is as follows:  
$TC\_SCP54[d]  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R W       |     |   | 5 |
| $TC\_SCP61 | REAL  | Offset for $TC\_DP11: $TC\_SCP61[t,d] analogous to $TC\_DP20[t,d]  
With active 'Flat D number management' function, the syntax is as follows:  
$TC\_SCP61[d]  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R W       |     |   | 5 |
| $TC\_SCP63 | REAL  | Offset for $TC\_DP3: $TC\_SCP63[t,d] analogous to $TC\_DP12[t,d]  
With active 'Flat D number management' function, the syntax is as follows:  
$TC\_SCP63[d]  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R W       |     |   | 5 |
| $TC\_SCP64 | REAL  | Offset for $TC\_DP4: $TC\_SCP64[t,d] analogous to $TC\_DP13[t,d]  
With active 'Flat D number management' function, the syntax is as follows:  
$TC\_SCP64[d]  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R W       |     |   | 5 |
| $TC\_SCP71 | REAL  | Offset for $TC\_DP11: $TC\_SCP71[t,d] analogous to $TC\_DP20[t,d]  
With active 'Flat D number management' function, the syntax is as follows:  
$TC\_SCP71[d]  
t: Tool number 1–32000  
d: Cutting edge number/D number 1–32000 | R W       |     |   | 5 |
## 15.2 List of System Variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_ECP13</td>
<td>REAL</td>
<td>Offset for $TC_DP3: $TC_ECP13[t,d] analogous to $TC_DP12[t,d]</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP13[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ECP14</td>
<td>REAL</td>
<td>Offset for $TC_DP4: $TC_ECP14[t,d] analogous to $TC_DP13[t,d]</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP14[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ECP21</td>
<td>REAL</td>
<td>Offset for $TC_DP11: $TC_ECP21[t,d] analogous to $TC_DP20[t,d]</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP21[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ECP23</td>
<td>REAL</td>
<td>Offset for $TC_DP3: $TC_ECP23[t,d] analogous to $TC_DP12[t,d]</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP23[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ECP24</td>
<td>REAL</td>
<td>Offset for $TC_DP4: $TC_ECP24[t,d] analogous to $TC_DP13[t,d]</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP24[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ECP31</td>
<td>REAL</td>
<td>Offset for $TC_DP11: $TC_ECP31[t,d] analogous to $TC_DP20[t,d]</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP31[d]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t: Tool number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d: Cutting edge number/D number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td>----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$TC_ECP33$</td>
<td>REAL</td>
<td>Offset for $TC_DP3$: $TC_ECP33[t,d]$ analogous to $TC_DP12[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_ECP33[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP34$</td>
<td>REAL</td>
<td>Offset for $TC_DP4$: $TC_ECP34[t,d]$ analogous to $TC_DP13[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_ECP34[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP41$</td>
<td>REAL</td>
<td>Offset for $TC_DP11$: $TC_ECP41[t,d]$ analogous to $TC_DP20[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_ECP41[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP43$</td>
<td>REAL</td>
<td>Offset for $TC_DP3$: $TC_ECP43[t,d]$ analogous to $TC_DP12[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_ECP43[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP44$</td>
<td>REAL</td>
<td>Offset for $TC_DP4$: $TC_ECP44[t,d]$ analogous to $TC_DP13[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_ECP44[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP51$</td>
<td>REAL</td>
<td>Offset for $TC_DP11$: $TC_ECP51[t,d]$ analogous to $TC_DP20[t,d]$ With active 'Flat D number management' function, the syntax is as follows: $TC_ECP51[d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
## 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_ECP53</td>
<td>REAL</td>
<td>Offset for $TC_DP3: $TC_ECP53[t,d] analogous to $TC_DP12[t,d] With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP53[d] t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP54</td>
<td>REAL</td>
<td>Offset for $TC_DP4: $TC_ECP54[t,d] analogous to $TC_DP13[t,d] With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP54[d] t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP61</td>
<td>REAL</td>
<td>Offset for $TC_DP11: $TC_ECP61[t,d] analogous to $TC_DP20[t,d] With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP61[d] t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP63</td>
<td>REAL</td>
<td>Offset for $TC_DP3: $TC_ECP63[t,d] analogous to $TC_DP12[t,d] With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP63[d] t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP64</td>
<td>REAL</td>
<td>Offset for $TC_DP4: $TC_ECP64[t,d] analogous to $TC_DP13[t,d] With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP64[d] t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_ECP71</td>
<td>REAL</td>
<td>Offset for $TC_DP11: $TC_ECP71[t,d] analogous to $TC_DP20[t,d] With active ‘Flat D number management’ function, the syntax is as follows: $TC_ECP71[d] t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
15.2.8 Monitoring data for tool management

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_MOP1$</td>
<td>REAL</td>
<td>$TC_MOP1[t,d]$ Prewarning limit for tool life t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP2$</td>
<td>REAL</td>
<td>$TC_MOP2[t,d]$ Remaining tool life t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP3$</td>
<td>INT</td>
<td>$TC_MOP3[t,d]$ Prewarning limit for number of workpieces t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP4$</td>
<td>INT</td>
<td>$TC_MOP4[t,d]$ Remaining number of workpieces t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP5$</td>
<td>REAL</td>
<td>$TC_MOP5[t,d]$ Prewarning limit wear t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP6$</td>
<td>REAL</td>
<td>$TC_MOP6[t,d]$ Remaining wear t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP11$</td>
<td>REAL</td>
<td>$TC_MOP11[t,d]$ Service life setpoint t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP13$</td>
<td>INT</td>
<td>$TC_MOP13[t,d]$ Workpiece count setpoint t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_MOP15$</td>
<td>REAL</td>
<td>$TC_MOP15[t,d]$ Wear setpoint t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.9 Monitoring data for OEM users

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_MOPC1$</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MOPC1[t,d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$TC_MOPC2$</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MOPC2[t,d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_MOPC10$</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MOPC10[t,d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$TC_MOPCS1$</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MOPCS1[t,d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$TC_MOPCS2$</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MOPCS2[t,d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_MOPCS10$</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MOPCS10[t,d]$ t: Tool number 1–32000 d: Cutting edge number/D number 1–32000</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
### 15.2.10 Tool-related data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $TC_TP1    | INT    | $TC_TP1[t]
Duplo number
 t: Tool number 1–32000                                       | R W        |    |   | 2 |
| $TC_TP2    | STRING | $TC_TP2[t]
Tool name
 t: Tool number 1–32000
2nd dimension for TYPE_STRING is automatically
 MAXSTRINGLEN                                      | R W        |    |   | 2 |
| $TC_TP3    | INT    | $TC_TP3[t]
Size to left
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP4    | INT    | $TC_TP4[t]
Size to right
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP5    | INT    | $TC_TP5[t]
Size toward top
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP6    | INT    | $TC_TP6[t]
Size toward bottom
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP7    | INT    | $TC_TP7[t]
Magazine location type
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP8    | INT    | $TC_TP8[t]
Status
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP9    | INT    | $TC_TP9[t]
Type of tool monitoring
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP10   | INT    | $TC_TP10[t]
Tool info
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TP11   | INT    | $TC_TP11[t]
Replacement strategy
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
| $TC_TPC1   | REAL   | The type can be defined in the machine data.
The default is INT
 $TC_TPC1[t]
 t: Tool number 1–32000                                      | R W        |    |   | 2 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC}_{\text{TPC}2}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is INT $\text{TC}_{\text{TPC}2}[t]$ $t$: Tool number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC}_{\text{TPC}10}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is INT $\text{TC}_{\text{TPC}10}[t]$ $t$: Tool number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$\text{TC}_{\text{TPCS}1}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is INT $\text{TC}_{\text{TPCS}1}[t]$ $t$: Tool number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\text{TC}_{\text{TPCS}2}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is INT $\text{TC}_{\text{TPCS}2}[t]$ $t$: Tool number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$\text{TC}_{\text{TPCS}10}$</td>
<td>REAL</td>
<td>The type can be defined in the machine data. The default is INT $\text{TC}_{\text{TPCS}10}[t]$ $t$: Tool number 1–32000</td>
<td>R</td>
<td>W</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.11 Tool-related grinding data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_TPG1}$</td>
<td>INT</td>
<td>$\text{TC_TPG1}[t]$ Spindle number t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG2}$</td>
<td>INT</td>
<td>$\text{TC_TPG2}[t]$ Chaining rule t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG3}$</td>
<td>REAL</td>
<td>$\text{TC_TPG3}[t]$ Minimum grinding wheel radius t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG4}$</td>
<td>REAL</td>
<td>$\text{TC_TPG4}[t]$ Minimum grinding wheel width t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG5}$</td>
<td>REAL</td>
<td>$\text{TC_TPG5}[t]$ Current grinding wheel width t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG6}$</td>
<td>REAL</td>
<td>$\text{TC_TPG6}[t]$ Maximum rotation speed t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG7}$</td>
<td>REAL</td>
<td>$\text{TC_TPG7}[t]$ Maximum surface speed t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG8}$</td>
<td>REAL</td>
<td>$\text{TC_TPG8}[t]$ Inclination angle for oblique grinding wheel t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_TPG9}$</td>
<td>INT</td>
<td>$\text{TC_TPG9}[t]$ Parameter number for radius calculation t: Tool number 1–32000</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 15.2.12 Magazine location data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_MPP1</td>
<td>INT</td>
<td>$TC_MPP1[n,m] Location class n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP2</td>
<td>INT</td>
<td>$TC_MPP2[n,m] Location type n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP3</td>
<td>BOOL</td>
<td>$TC_MPP3[n,m] Adjacent location consideration on/off n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP4</td>
<td>INT</td>
<td>$TC_MPP4[n,m] Location status n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP5</td>
<td>INT</td>
<td>$TC_MPP5[n,m] Buffer magazine: Location class index Real magazines: Wear group number n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP6</td>
<td>INT</td>
<td>$TC_MPP6[n,m] T-no. of the tool at this location n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP7</td>
<td>INT</td>
<td>$TC_MPP7[n,m] Adapter number of tool adapter at this location n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$TC_MPP66</td>
<td>INT</td>
<td>$TC_MPP66[n,m] T-no. of the tool in the buffer, for which the location specified by n,m is reserved. A write operation only makes sense when loading a backup file to the NCK. Name assignment follows the $TC_MPP6 - tool no. of the tool at the magazine location. n: Physical magazine number: m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.13 Magazine location data for OEM users

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_MPPC1</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MPPC1[n,m] n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPPC2</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MPPC2[n,m] n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_MPPC10</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MPPC10[n,m] n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MPPCS1</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MPPCS1[n,m] n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$TC_MPPCS2</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MPPCS2[n,m] n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_MPPCS10</td>
<td>INT</td>
<td>The type can be defined in the machine data. The default is INT $TC_MPPCS10[n,m] n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$TC_MDP1</td>
<td>INT</td>
<td>$TC_MDP1[n,m] Distance between change position of magazine n and location m of the 1st internal magazine internal mag. 1 distance parameter n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$TC_MDP2</td>
<td>INT</td>
<td>$TC_MDP2[n,m] Distance between change position of magazine n and location m of the 2nd internal magazine internal mag. 2 distance parameter n: Physical magazine number; m: Physical location number</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $TC_MLSR  | INT  | $TC_MLSR[n,m]=0  
Assignment between buffer location n and buffer location m
m must identify a location of type ‘spindle’.
n must identify a location not of type ‘spindle’.
This can be used to define, for example, which grippers,... are assigned to which spindles. The value for parameter value is defined as fix = 0.
The write process defines a relation, the read process checks whether a particular relation applies. If not, an alarm is produced during a read operation.
define links of grippers,... to spindles.
n: Physical magazine location number of location class not equal to SPINDLE
m: Physical magazine location number of location class equal to SPINDLE | R W | 3 |
| $TC_MPTH  | INT  | $TC_MPTH[n,m]  
Magazine location type hierarchy
mag. location (place)types hierarchy parameter
n: Hierarchy 0 - 8-1
m: Location type 0 - 8 - 1 | R W | 3 |

### 15.2.14 Magazine description data for tool management

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $TC_MAP1  | INT  | $TC_MAP1[n]  
Type of magazine
n: Magazine number 1 to ... | R W | 2 |
| $TC_MAP2  | STRING | $TC_MAP2[n]  
Identifier of the magazine
n: Magazine number 1 to ...
2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN | R W | 2 |
| $TC_MAP3  | INT  | $TC_MAP3[n]  
State of magazine
n: Magazine number 1 to ... | R W | 2 |
| $TC_MAP4  | INT  | $TC_MAP4[n]  
Chaining with following magazine
n: Magazine number 1 to ... | R W | 2 |
| $TC_MAP5  | INT  | $TC_MAP5[n]  
Chaining with previous magazine
n: Magazine number 1 to ... | R W | 2 |
| $TC_MAP6  | INT  | $TC_MAP6[n]  
Number of rows
n: Magazine number 1 to ... | R W | 2 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_MAP7}$</td>
<td>INT</td>
<td>$\text{TC_MAP7}[n]$ (Number of columns, n: Magazine number 1 to ... )</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_MAP8}$</td>
<td>INT</td>
<td>$\text{TC_MAP8}[n]$ (Current magazine position with reference to the change position, n: Magazine number 1 to ... )</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_MAP9}$</td>
<td>INT</td>
<td>$\text{TC_MAP9}[n]$ (Current wear group number, n: Magazine number 1 to ... )</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_MAP10}$</td>
<td>INT</td>
<td>$\text{TC_MAP10}[n]$ (Current magazine search strategies. - tool search strategy - empty location search strategy The default entered by the NCK is the value $\text{TC_MAP2}$. n: Magazine number 1 to ...)</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

#### 15.2.15 Tool management magazine description data for OEM users

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_MAPC1}$</td>
<td>INT</td>
<td>$\text{TC_MAPC1}[n]$ (The type can be defined in the machine data. The default is INT)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\text{TC_MAPC2}$</td>
<td>INT</td>
<td>$\text{TC_MAPC2}[n]$ (The type can be defined in the machine data. The default is INT)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_MAPC10}$</td>
<td>INT</td>
<td>$\text{TC_MAPC10}[n]$ (The type can be defined in the machine data. The default is INT)</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{TC_MAPCS1}$</td>
<td>INT</td>
<td>$\text{TC_MAPCS1}[n]$ (The type can be defined in the machine data. The default is INT)</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$\text{TC_MAPCS2}$</td>
<td>INT</td>
<td>$\text{TC_MAPCS2}[n]$ (The type can be defined in the machine data. The default is INT)</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{TC_MAPCS10}$</td>
<td>INT</td>
<td>$\text{TC_MAPCS10}[n]$ (The type can be defined in the machine data. The default is INT)</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved

Siemens AG, 2002. All rights reserved

Siemens AG, 2002. All rights reserved
### 15.2.16 Magazine module parameter

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_MAMP1</td>
<td>STRING</td>
<td>Identifier of the magazine module</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scalar variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd dimension for TYPE_STRING is automatically</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAXSTRINGLEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_MAMP2</td>
<td>INT</td>
<td>Type of tool search</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scalar variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_MAMP3</td>
<td>INT</td>
<td>Handling of tools with wear groups</td>
<td>R W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scalar variable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 15.2.17 Adapter data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TC_ADPTT</td>
<td>INT</td>
<td>Adapter transformation number</td>
<td>R W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Adapter number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ADPT1</td>
<td>REAL</td>
<td>Adapter geometry: Length 1</td>
<td>R W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Adapter number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ADPT2</td>
<td>REAL</td>
<td>Adapter geometry: Length 2</td>
<td>R W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Adapter number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$TC_ADPT3</td>
<td>REAL</td>
<td>Adapter geometry: Length 3</td>
<td>R W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Adapter number 1–32000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 15.2.18 Measuring system compensation values

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_ENC_COMP_P</td>
<td>REAL</td>
<td>$AA_ENC_COMP_P[n,m,a] Compensation values</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Encoder no. 0–1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m: Point no. 0–&lt;MD value&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: Machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_ENC_COMP_STEP</td>
<td>REAL</td>
<td>$AA_ENC_COMP_STEP[n,a] Step width</td>
<td>R W</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Encoder no. 0–1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: Machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\$AA\_ENC\_COM P\_MIN | REAL   | $AA\_ENC\_COMP\_MIN[n,a]$Compensation start position  
a: Machine axis  
n: Encoder no. 0–1  
Axes: Machine axis | R         | W              | 2  |    |
| $\$AA\_ENC\_COM P\_MAX  | REAL   | $AA\_ENC\_COMP\_MAX[n,a]$Compensation end position  
a: Machine axis  
n: Encoder no. 0–1  
Axes: Machine axis | R         | W              | 2  |    |
| $\$AA\_ENC\_COM P\_IS\_MODULO | BOOL   | $AA\_ENC\_COMP\_IS\_MODULO[n,a]$Compensation is modulo  
a: Machine axis  
n: Encoder no. 0–1  
Axes: Machine axis | R         | W              | 2  |    |

#### 15.2.19 Quadrant error compensation

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\$AA\_QEC     | REAL   | $AA\_QEC[n,m,a]$Result of learning process  
a: Machine axis  
n: 0  
m: No. of point: 0 - $MN\_MM\_QEC\_MAX\_POINTS  
Axes: Machine axis | R         | W              | 2  |    |
| $\$AA\_QEC\_COARSE\_STEPS | INT   | $AA\_QEC\_COARSE\_STEPS[n,a]$Compensation value: Coarse quantization of the characteristic  
a: Machine axis  
n: 0  
Axes: Machine axis | R         | W              | 2  |    |
| $\$AA\_QEC\_FINE\_STEPS | INT   | $AA\_QEC\_FINE\_STEPS[n,a]$Fine quantization of characteristic  
a: Machine axis  
n: 0  
Axes: Machine axis | R         | W              | 2  |    |
| $\$AA\_QEC\_ACCEL\_1 | REAL   | $AA\_QEC\_ACCEL\_1[n,a]$Acceleration in 1st knee-point according to definition [mm/s² o. inch/s² o. degrees/s²]  
a: Machine axis  
n: 0  
Axes: Machine axis | R         | W              | 2  |    |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AA_QEC_ACCEL_2   | REAL  | $AA_QEC_ACCEL_2[n,a]
Acceleration in 2nd knee-point according to definition [mm/s² o. inch/s² o. degrees/s²]
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_ACCEL_3   | REAL  | $AA_QEC_ACCEL_3[n,a]
Acceleration in 3rd knee-point according to definition [mm/s² o. inch/s² o. degrees/s²]
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_MEAS_TIME_1| REAL | $AA_QEC_MEAS_TIME_1[n,a]
Measuring time for the range $AA_QEC_ACCEL_1
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_MEAS_TIME_2| REAL | $AA_QEC_MEAS_TIME_2[n,a]
Measuring time for the range $AA_QEC_ACCEL_2
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_MEAS_TIME_3| REAL | $AA_QEC_MEAS_TIME_3[n,a]
Measuring time for the range $AA_QEC_ACCEL_3
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_TIME_1     | REAL  | $AA_QEC_TIME_1[n,a]
1st filter time for feedforward element
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_TIME_2     | REAL  | $AA_QEC_TIME_2[n,a]
2nd filter time for feedforward element
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_LEARNING_RATE| REAL | $AA_QEC_LEARNING_RATE[n,a]
Learning rate for network
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
| $AA_QEC_DIRECTIONAL| BOOL | $AA_QEC_DIRECTIONAL[n,a]
TRUE: Compensation is directional
FALSE: Compensation is not directional
a: Machine axis
n: 0
Axes: Machine axis | R        | W    | 2 |
### 15.2.20 Interpolatory compensation

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AN_CEC}$</td>
<td>REAL</td>
<td>$\text{AN_CEC}[n,m]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compensation value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m: No. of interpolation point 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_INPUT_AXIS}$</td>
<td>AXIS</td>
<td>$\text{AN_CEC_INPUT_AXIS}[n]$: Name of axis whose setpoint is to act as the compensation table input</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_OUTPUT_AXIS}$</td>
<td>AXIS</td>
<td>$\text{AN_CEC_OUTPUT_AXIS}[n]$: Name of axis which is influenced by the compensation table output</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v$\text{AN_CEC_STEP}$</td>
<td>REAL</td>
<td>$\text{AN_CEC_STEP}[n]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distance between compensation values</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_MIN}$</td>
<td>REAL</td>
<td>$\text{AN_CEC_MIN}[n]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start position of compensation table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_MAX}$</td>
<td>REAL</td>
<td>$\text{AN_CEC_MAX}[n]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End position of compensation table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_DIRECTION}$</td>
<td>INT</td>
<td>$\text{AN_CEC_DIRECTION}[n]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activates directional action of compensation table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_MULT_BY_TABLE}$</td>
<td>INT</td>
<td>$\text{AN_CEC_MULT_BY_TABLE}[n]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of table for which the initial value is to be multiplied by the initial value of the compensation table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Both traversing directions of basic axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Positive traversing direction of basic axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1: Negative traversing direction of basic axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AN_CEC_IS_MODULO}$</td>
<td>BOOL</td>
<td>$\text{AN_CEC_IS_MODULO}[n]$</td>
<td></td>
<td>R</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: Cyclic repetition of compensation table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: No cyclic repetition of compensation table</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: No. of compensation table 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.21 NCK-specific protection zones

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_{PA_ACTIV_IMMED}$</td>
<td>BOOL</td>
<td>Protection zone active immediately?</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: The protection zone is active immediately once the control is powered up and the axes are referenced</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: The protection zone is not active immediately</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_{PA_T_W}$</td>
<td>CHAR</td>
<td>Part/tool related protection zone</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Part-related protection zone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Tool-related protection zone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_{PA_ORI}$</td>
<td>INT</td>
<td>Orientation of protection zone</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Polygon in plane from 1st and 2nd geo axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Polygon in plane from 3rd and 1st geo axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Polygon in plane from 2nd and 3rd geo axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_{PA_LIM_3_DIM}$</td>
<td>INT</td>
<td>Code for restricting the protection zone in the axis that lies perpendicular to the polygon definition</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: = No limit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: = Limit in positive direction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: = Limit in negative direction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: = Limit in both directions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_{PA_PLUS_LIM}$</td>
<td>REAL</td>
<td>Positive limit of the protection zones in the axis that lies perpendicular to the polygon definition</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_{PA_MINUS_LIM}$</td>
<td>REAL</td>
<td>Negative limitation of protection zone in the negative direction in the axis that lies perpendicular to the polygon definition</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SN_{PA_CONT_NUM}$</td>
<td>INT</td>
<td>Number of valid contour elements</td>
<td>R</td>
<td>W</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n ): Number of protection zone 0 - (maximum value settable via MD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SN_PA_CONT_TYP</td>
<td>INT</td>
<td>$SN_PA_CONT_TYP[n,m] Contour element type (G1, G2, G3) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CONT_ORD</td>
<td>REAL</td>
<td>$SN_PA_CONT_ORD[n,m] End point of contour element (ordinate) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS</td>
<td>REAL</td>
<td>$SN_PA_CONT_ABS[n,m] End point of contour element (abscissa) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD</td>
<td>REAL</td>
<td>$SN_PA_CENT_ORD[n,m] Center point of contour element (ordinate) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS</td>
<td>REAL</td>
<td>$SN_PA_CENT_ABS[n,m] Center point of contour element (abscissa) n: Number of protection zone 0 - (maximum value settable via MD) m: Number of contour element 0–10</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

15.2.22 Cycle parameterization

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_A</td>
<td>REAL</td>
<td>$C_A Value of programmed address A in Fanuc mode for cycle parameterization</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>$C_B</td>
<td>REAL</td>
<td>$C_B Value of programmed address B in Fanuc mode for cycle parameterization</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_H</td>
<td>REAL</td>
<td>$C_H Value of programmed address H in Fanuc mode for cycle parameterization</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$C_I$</td>
<td>REAL</td>
<td>$C_I[]$ Value of programmed address I in Fanuc mode for cycle parameterization and macro technique with G65/G66. For macro programming with G65/G66, up to 10 entries are possible in the block with address I. The values are in the programmed sequence in the array.</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$C_J$</td>
<td>REAL</td>
<td>$C_J[]$ Value of programmed address J in Fanuc mode for cycle parameterization and macro technique with G65/G66. For macro programming with G65/G66, up to 10 entries are possible in the block with address J. The values are in the programmed sequence in the array.</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$C_K$</td>
<td>REAL</td>
<td>$C_K[]$ Value of programmed address K in Fanuc mode for cycle parameterization and macro technique with G65/G66. For macro programming with G65/G66, up to 10 entries are possible in the block with address K. The values are in the programmed sequence in the array.</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$C_L$</td>
<td>REAL</td>
<td>$C_L$ Value of programmed address L in Fanuc mode for cycle parameterization</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$C_M$</td>
<td>REAL</td>
<td>$C_M$ Value of programmed address M in Fanuc mode for cycle parameterization</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_Z$</td>
<td>REAL</td>
<td>$C_Z$ Value of programmed address Z in Fanuc mode for cycle parameterization</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$C_DL$</td>
<td>REAL</td>
<td>Value of programmed address DL (additive tool offset) for A subroutine call by M/T function replacement</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$C_TS$</td>
<td>STRING</td>
<td>$C_TS$ String of the tool identifier programmed under address T for tool function replacement (during active tool monitoring only)</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
</tbody>
</table>
## 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{C}_A_\text{PROG}$</td>
<td>INT</td>
<td>$\text{C}_A_\text{PROG}$ Address A is programmed to a block with cycle call 0 = not programmed 1 = programmed 3 = programmed as incremental Bit 0 / value 1 is set when the address is programmed as absolute or incremental. To distinguish between absolute and incremental, bit 1 / value 3 is also set. The bit 2 =0 value is programmed as INT the =1 value is programmed as REAL</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$\text{C}_B_\text{PROG}$</td>
<td>INT</td>
<td>$\text{C}_B_\text{PROG}$ Address B is programmed to a block with cycle call 0 = not programmed 1 = programmed 3 = programmed as incremental Bit 0 / value 1 is set when the address is programmed as absolute or incremental. To distinguish between absolute and incremental, bit 1 / value 3 is also set. The bit 2 =0 value is programmed as INT the =1 value is programmed as REAL</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{C}_Z_\text{PROG}$</td>
<td>INT</td>
<td>$\text{C}_Z_\text{PROG}$ Address Z is programmed to a block with cycle call 0 = not programmed 1 = programmed 3 = programmed as incremental Bit 0 / value 1 is set when the address is programmed as absolute or incremental. To distinguish between absolute and incremental, bit 1 / value 3 is also set. The bit 2 =0 value is programmed as INT the =1 value is programmed as REAL</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$\text{C}_\text{DL}_\text{PROG}$</td>
<td>INT</td>
<td>Queries whether during a subroutine call by M/T function replacement the address DL (additive tool offset) has been programmed. 0 = not programmed 1 = An additive tool offset has been programmed under the address DL.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>.</td>
</tr>
</tbody>
</table>
### List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_TS_PROG</td>
<td>INT</td>
<td>Queries whether, in the case of a subroutine call by T function replacement a tool identifier has been programmed under address T. (with active tool monitoring only) 0 = not programmed 1 = programmed</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$C_ALL_PROG</td>
<td>INT</td>
<td>$C_ALL_PROG Bit pattern of all the programmed addresses in a block with cycle call bit0 = address &quot;A&quot; ... bit25 = address &quot;Z&quot; bit = 1 -&gt; address programmed bit = 0 -&gt; address not programmed</td>
<td>R</td>
<td>W</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$C_INC_PROG</td>
<td>INT</td>
<td>$C_INC_PROG Bit pattern of all the addresses programmed as incremental in a block with cycle call bit0 = address &quot;A&quot; ... bit25 = address &quot;Z&quot; bit = 1 -&gt; address programmed as incremental bit = 0 -&gt; address programmed as absolute</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$C_TYP_PROG</td>
<td>INT</td>
<td>$C_TYP_PROG Bit pattern of all the programmed addresses with the value INT or REAL bit0 = address &quot;A&quot; ... bit25 = address &quot;Z&quot; Bit = 1 -&gt; address programmed with real value Bit = 0 -&gt; address programmed with int value</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$C_I_NUM</td>
<td>INT</td>
<td>$C_I_NUM contains the number of I addresses programmed in the block. For cycle programming, this value is always 1 whenever bit 0 in $C_I_PROG is set. In the case of macro programming with G65/G66, this contains the number of &quot;I&quot; addresses programmed in the block, (max. 10).</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$C_J_NUM</td>
<td>INT</td>
<td>$C_J_NUM contains the number of J addresses programmed in the block. For cycle programming, this value is always 1 whenever bit 0 in $C_J_PROG is set. In the case of macro programming with G65/G66, this contains the number of &quot;J&quot; addresses programmed in the block, (max. 10).</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$C_K_NUM</td>
<td>INT</td>
<td>$C_K_NUM contains the number of K addresses programmed in the block. For cycle programming, this value is always 1 whenever bit 0 in $C_K_PROG is set. In the case of macro programming with G65/G66, this contains the number of &quot;K&quot; addresses programmed in the block, (max. 10).</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\$C_I\_ORDER | INT    | $\$C_I\_ORDER[]
Number of the IJK block, in which I has been programmed
For macro programming with G65/G66, up to 10 entries are possible in the block with address I. These can be used to evaluate the IJK sequence. A note is always made of which IJK go together. | R         | W   | 6 | .  |
| $\$C_J\_ORDER | INT    | $\$C_J\_ORDER[]
Number of the IJK block, in which J has been programmed
For macro programming with G65/G66, up to 10 entries are possible in the block with address J. These can be used to evaluate the IJK sequence.
from the parts program. A note is always made of which IJK go together. | R         | W   | 6 | .  |
| $\$C_K\_ORDER | INT    | $\$C_K\_ORDER[]
Number of the IJK block, in which K has been programmed
For macro programming with G65/G66, up to 10 entries are possible in the block with address K. These can be used to evaluate the IJK sequence from the parts program. A note is always made of which IJK go together. | R         | W   | 6 | .  |
| $\$C_ME | INT    | $\$C_ME
Address extension for address M in the case of a subroutine call by M function | R         | W   | 6 | .  |
| $\$C_TE | INT    | $\$C_TE
Address extension for address T in the case of a subroutine call by T function | R         | W   | 6 | .  |
| $\$C_MACPAR | REAL   | $\$MAC\_PAR[n]
Macro variable in ISO2/3 mode programmed in the original program with
#<Number>
The max. number of ISO macro parameters is 33 | R         | W   | 6 | .  |
## 15.2.23 System data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{SAN_SETUP_TIME}$</td>
<td>REAL</td>
<td>IF $\text{SAN_SETUP_TIME} &gt; 60000$ GOTOF MARK01 Time since last power up of control with default values (in minutes)</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAN_POWERON_TIME}$</td>
<td>REAL</td>
<td>IF $\text{SAN_POWERON_TIME} == 480$ GOTOF MARK02 Time since last power-on of control (in minutes)</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAN_NCK_VERSION}$</td>
<td>REAL</td>
<td>NCK version: only the part of the floating-point number prior to the decimal point is evaluated, the part after the decimal point can contain identification for intermediate states within development. The part prior to the decimal point contains the official software version identifier of the NCK: For example, if 20.00.00 is for the NCK version, the value of the variable is 200000.0 compare OPI N/Y nckVersion</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 15.2.24 Frames 2

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{SP_UBFR}$</td>
<td>FRAME</td>
<td>$\text{SP_UBFR}$ 1st base frame in channel activated after G500, G54..G599. Corresponds to $\text{SP_CHBFR}[0]$. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SP_SETFRAME}$</td>
<td>FRAME</td>
<td>$\text{SP_SETFRAME}$ Current system frame for preset actual value memory and scratching. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SP_EXTFRAME}$</td>
<td>FRAME</td>
<td>$\text{SP_EXTFRAME}$ Current system frame for zero offset external. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SP_PARTFRAME}$</td>
<td>FRAME</td>
<td>$\text{SP_PARTFRAME}$ Current system frame for TCARR and PAROT. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SP_TOOLFRAME}$</td>
<td>FRAME</td>
<td>$\text{SP_TOOLFRAME}$ Current system frame for TOROT and TOFRAME. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SP_WPFRAME}$</td>
<td>FRAME</td>
<td>$\text{SP_WPFRAME}$ Current system frame for part reference points. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SP_CYCFRAME}$</td>
<td>FRAME</td>
<td>$\text{SP_CYCFRAME}$ Current system frame for cycles. Axes: (geometry axis, channel axis, machine axis)</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| $P\_CHBFRAME  | FRAME     | $P\_CHBFRAME[n]  
Current base frames in the channel. Configurable via MD $MC\_MM\_NUM\_BASE\_FRAMES. The dimension is checked on variable access. Axes: (geometry axis, channel axis, machine axis) | R         | W   | 5 |   |
| $P\_NCBFRAME  | FRAME     | $P\_NCBFRAME[n]  
Current NCU base frames. Configurable via MD $MN\_MM\_NUM\_GLOBAL\_BASE\_FRAMES. The dimension is checked on variable access. Axes: (geometry axis, channel axis, machine axis) | R         | W   | 5 |   |
| $P\_ACTBFRAME | FRAME     | $P\_ACTBFRAME  
Current chained total basic frame Axes: (geometry axis, channel axis, machine axis) | R         |     |   | 5 |
| $P\_BFRAME    | FRAME     | $P\_BFRAME  
Current 1st base frame in the channel. Corresponds to $P\_CHBFRAME[0]. Axes: (geometry axis, channel axis, machine axis) | R         | W   | 4 |   |
| $P\_IFRAME    | FRAME     | $P\_IFRAME  
Current settable frame Axes: (geometry axis, channel axis, machine axis) | R         | W   | 2 |   |
| $P\_PFRA M    | FRAME     | $P\_PFRA M  
Current programmable frame Axes: (geometry axis, channel axis, machine axis) | R         | W   | 2 |   |
| $P\_ACTFRAME  | FRAME     | $P\_ACTFRAME  
Current total frame Axes: (geometry axis, channel axis, machine axis) | R         |     |   | 2 |
| $P\_UIFRNUM   | INT       | $P\_UIFRNUM  
Number of the active $P\_UIFR | R         |     | 2 |   |
| $P\_NCBFRMA K | INT       | $P\_NCBFRMA K  
Bit screenform is used for definition of the NCU global base frames that are included in the calculation of the total base frame. | R         | W   | 5 |   |
| $P\_CHBFRMA K | INT       | $P\_CHBFRMA K  
Bit screenform is used for definition of channel-specific base frames that are included in the calculation of the total base frame. | R         | W   | 5 |   |
### 15.2.25 Tool data

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_AD$</td>
<td>REAL</td>
<td>$P_AD[n]$ Active tool offsets n: Parameter number 1–31 n = 1-25 $TC_DP1$ to $TC_DP25$ n = 26 $TC_DPCE$ n = 27 $TC_DPH$ n = 28 $TC_DPV$ n = 29 $TC_DPV3$ n = 30 $TC_DPV4$ n = 31 $TC_DPV5$</td>
<td>R</td>
<td>W</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$P_ADT$</td>
<td>REAL</td>
<td>$P_ADT[n]$ With an active tool adapter, the transformed compensation values of the tool adapter transformation are returned when reading the values tool compensations are transformed n: Parameter number 1–31 n = 1-25 $TC_DP1$ to $TC_DP25$ n = 26 $TC_DPCE$ n = 27 $TC_DPH$ n = 28 $TC_DPV$ n = 29 $TC_DPV3$ n = 30 $TC_DPV4$ n = 31 $TC_DPV5$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$P_DLNO$</td>
<td>INT</td>
<td>$P_DLNO$ Active cumulative compensation number DL=0 - DL='max.'; 'max'= value of $MN_MM_MAX_SUMCORR_PER_CUTTEDGE$</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$P_TOOL$</td>
<td>INT</td>
<td>$P_TOOL$ Active tool cutting edge D0 - D'='max.'; 'max'= value of $MN_MM_MAX_CUTTING_EDGE_NO$</td>
<td>R</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_TOOLNO$</td>
<td>INT</td>
<td>$P_TOOLNO$ Active tool number T0 - T32000; with active function 'flat D number’ T can have 8 digits. The command should generally not be used when magazine management is active. When magazine management is active, GETEXET should be used instead. (The programming is only ever reliable in a situation where $MC_CUTTING_EDGE_DEFAULT=-1, &gt; 0$ applies. The wrong tool number can be determined for $MC_CUTTING_EDGE_DEFAULT=0, or =-2$. If programming takes place after programming D &gt; 0, it is also always reliable. NOTICE: Especially for $MC_CUTTING_EDGE_DEFAULT=-2, $P_TOOLNO$ (the tool no. of the active tool with which the currently effective D offset was being calculated) and GETEXET (the changed tool) can return different tool numbers. -&gt;also see $P_MTHSDC$ and documentation on the topic of more than one tool holder/spindle.</td>
<td>R</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$P_TOOLP$</td>
<td>INT</td>
<td>$P_TOOLP$ Last programmed tool number T0 - T32000 (for operation without magazine management). The command cannot be used when magazine management is active. When magazine management is active, GETSELT must be used instead.</td>
<td>R</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>$P_TOOLL$</td>
<td>REAL</td>
<td>$P_TOOLL[n]$ Active total tool length n: Length 1–3</td>
<td>R</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$P_TOOLO$</td>
<td>REAL</td>
<td>$P_TOOLO[n]$ Active tool orientation. n: Component 1–3</td>
<td>R</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>$AC_TOOLO_A CT$</td>
<td>REAL</td>
<td>$AC_TOOLO_ACT[n]$ Active setpoint orientation. n: Component 1–3</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>$AC_TOOLO_E ND$</td>
<td>REAL</td>
<td>$AC_TOOLO_END[n]$ End orientation of the active block n: Component 1–3</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>$AC_TOOLO_D IFF$</td>
<td>REAL</td>
<td>$AC_TOOLO_DIFF$ Residual angle of tool orientation in the active block</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>$VC_TOOLO$</td>
<td>REAL</td>
<td>$VC_TOOLO[n]$ Actual orientation n: Component 1–3</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn O S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$VC_TOOLO_DIFF</td>
<td>REAL</td>
<td>Angle between setpoint orientation and actual orientation</td>
<td>RS</td>
<td>R 6 . 4</td>
</tr>
<tr>
<td>$VC_TOOLO_STAT</td>
<td>INT</td>
<td>Status of the calculation of actual orientation</td>
<td>RS</td>
<td>R 6 . 4</td>
</tr>
<tr>
<td>$P_TC</td>
<td>INT</td>
<td>Active tool carrier</td>
<td>R</td>
<td>6 . 1</td>
</tr>
<tr>
<td>$AC_TC</td>
<td>INT</td>
<td>Active tool carrier</td>
<td>RS</td>
<td>R 6 . 4</td>
</tr>
<tr>
<td>$P_TCANG</td>
<td>REAL</td>
<td>Active angle of toolholder axis</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Angle 1–2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_TCDIFF</td>
<td>REAL</td>
<td>The difference between the calculated and the used angle of a toolholder axis when incrementing (Hirth tooth system) the angle</td>
<td>R</td>
<td>6 . 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Angle 1–2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_TCSOL</td>
<td>INT</td>
<td>Number of solutions when specifying the axis of rotation angle of an orientable toolholder from a frame With 0 to 2 solutions, the corresponding value is returned. With an infinite number of solutions, the return value is 3. If the angles are specified (TCOABS), the number of solutions is always 1.</td>
<td>R</td>
<td>6 . 1</td>
</tr>
<tr>
<td>$P_TCSTAT</td>
<td>INT</td>
<td>Specifies the status of an orientable toolholder. The variable is bit-coded with the following significance: 0x1 The first rotary axis is available 0x2 The second rotary axis is available 0x4 The angles used for the calculation come from an orientation in the frame direction 0x8 The angles used for the calculation have been specified as absolute 0x10 The pole axis angle is indeterminate for orientation in the frame direction 0x1000 Only the tool can be rotated (kinematics type T) 0x2000 Only the workpiece can be rotated (kinematics type P) 0x4000 Tool and workpiece can be rotated (kinematics type M) Bits not designated here are currently unassigned.</td>
<td>R</td>
<td>6 . 4</td>
</tr>
<tr>
<td>$P_TOOLR</td>
<td>REAL</td>
<td>Active tool radius (total)</td>
<td>R</td>
<td>2</td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
</tr>
<tr>
<td>$P_TOOLND</td>
<td>INT</td>
<td>$P_TOOLND[t]$ Number of cutting edges of tool t t: Tool number 1–32000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$P_TOOLEXIST</td>
<td>BOOL</td>
<td>$P_TOOLEXIST[t]$ Tool with T No. t exists t: Tool number 1–32000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$P_D</td>
<td>INT</td>
<td>$P_D$ Current D number in ISO_2-language mode</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$P_H</td>
<td>INT</td>
<td>$P_H$ Current H number in ISO_2-language mode</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$A_TOOLMN</td>
<td>INT</td>
<td>$A_TOOLMN[t]$ Magazine number of tool t t: Tool number 1–32000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$A_TOOLMLN</td>
<td>INT</td>
<td>$A_TOOLMLN[t]$ Magazine location number of tool t t: Tool number 1–32000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$A_MYMN</td>
<td>INT</td>
<td>$A_MYMN[t]$ Owner magazine number of the tool with T number t. Result value = 0 = tool is not loaded (if $A_TOOLMN &gt; 0$, then manual tool). Result value = -1 = tool management is not active Result value = -2 = tool with T number t does not exist. t: Tool number 1–32000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$A_MYMLN</td>
<td>INT</td>
<td>$A_MYMLN[t]$ Owner magazine location number of the tool with T number t. Result value = 0 = tool is not loaded (if $A_TOOLMLN &gt; 0$, then manual tool). Result value = -1 = tool management is not active Result value = -2 = tool with T number t does not exist. t: Tool number 1–32000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>$A_MONIFACT</td>
<td>REAL</td>
<td>$A_MONIFACT$ Factor for tool length monitoring</td>
<td>R</td>
<td>WS</td>
</tr>
</tbody>
</table>
### List of System Variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts Pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{TOOLNG}$</td>
<td>INT</td>
<td>$P_{TOOLNG}$&lt;br&gt;Number of defined tool groups assigned to the channel&lt;br&gt;OPI module type = TM</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$P_{TOOLNT}$</td>
<td>INT</td>
<td>$P_{TOOLNT}$&lt;br&gt;Number of defined tools assigned to the channel&lt;br&gt;OPI module type = TV</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
</tbody>
</table>
| $P_{TOOLT}$ | INT | $P_{TOOLT}[i]$
  - i-th tool number T
  - OPI module type = TV
  - i=1,....,$P_{TOOLNT}$ | R | 6 | . | 1 |
| $P_{TOOLD}$ | INT | $P_{TOOLD}[t,i]$
  - i-th D-no of the tool with T number t; i=1,2...
  - if t is the value of a non-defined tool, -2 is returned
  - if i is a value outside the permitted range, 0 is returned.
  - OPI module type = TO
  - t = 1, ..... 32000
  - i = 1,....,$P_{TOOLND}$ | R | 6 | . | 1 |
| $P_{USEKT}$ | INT | $P_{USEKT}$ (= USE Kind of Tool)<br>Is a bit-coded value<br>All the tools whose parameter $TC_{TP11}$ has set one of the bits of $P_{USEKT},$<br>are available to the following tool changes. The value zero has the same content as 'all bits set'
  - OPI module = C/S | R | W | 6 | . | 1 |
| $P_{TOOLNDL}$ | INT | $P_{TOOLNDL}[t,d]$
  - Number of DL offsets of the D offset given by T number t and D number d
  - >0 Number of DL offsets
  - 0 no DL offset for this D offset
  - -1 sum offset function not active
  - -2 t is the value of a non-defined tool
  - -3 d is the value of a non-defined D offset
  - OPI module type = TO memory; TO unit
  - t = 1, ..... 32000
  - d = 1,...., 32000 | R | 6 | . | 1 |
### 15.2.26 Magazines

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\mathbf{P\_MAGN}$ | INT   | $\mathbf{P\_MAGN}$
Number of defined magazines assigned to the channel
- $> 0$ successful read access
- $0$ no magazines defined
- $-1$ TMMG not active
OPI module = TM | R |   | 6 | 1 |
| $\mathbf{P\_MAG}$ | INT   | $\mathbf{P\_MAG}[i]$
$i$-th magazine number
- $> 0$ successful read access
- $0$ $i$ is outside the permitted range
- $-1$ TMMG not active
OPI module = TM
$i = 1,..., P\_MAGN$ | R |   | 6 | 1 |
| $\mathbf{P\_MAGNDIS}$ | INT   | $\mathbf{P\_MAGNDIS}[n, m]$
Number of magazines interconnected with location $m$ of the internal magazine $n$.
- $> 0$ successful read access
- $0$ no magazine interconnected with the buffer location
- $-1$ TMMG not active
- $-2$ $n$ is not the number of an internal magazine
- $-3$ $m$ is not the number of an internal magazine location
OPI module = TPM
$n$ must be the number of the buffer magazine or of the loading magazine
$m = 1,..., \text{max. number of a location in the internal magazine mentioned}$ | R |   | 6 | 1 |
| $\mathbf{P\_MAGDISS}$ | INT   | $\mathbf{P\_MAGDISS}[l, i]$
Number of the $i$-th magazine interconnected with location $l$ of the buffer magazine.
- $> 0$ successful read access
- $0$ $i$ is outside the permitted range
- $-1$ TMMG not active
- $-2$ $m$ is not the number of a buffer location
- $-3$ no buffer magazine defined
OPI module = TPM
$l = 1,..., \text{max. number of a location in the buffer magazine}$ | R |   | 6 | 1 |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{MAGDISL}}$</td>
<td>INT</td>
<td>$P_{\text{MAGDISL}}[i, l]$&lt;br&gt;Number of the i-th magazine interconnected with location l of the loading magazine.&lt;br&gt;$&gt;0$ successful read access&lt;br&gt;$0$ i is outside the permitted range&lt;br&gt;$-1$ TMMG not active&lt;br&gt;$-2$ m is not the number of a loading magazine location&lt;br&gt;$-3$ no loading magazine defined&lt;br&gt;OPI module = TPM&lt;br&gt;$l=1,...,\text{max. number of a location in the loading magazine}$&lt;br&gt;$i=1,...,#P_{\text{MAGDISL}}[\text{no. of the loading magazine, refLoc }]$</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$P_{\text{MAGNS}}$</td>
<td>INT</td>
<td>$P_{\text{MAGNS}}$&lt;br&gt;Number of spindle locations / toolholder locations in the buffer assigned to the channel.&lt;br&gt;$&gt;0$ successful read access&lt;br&gt;$0$ no spindle locations defined&lt;br&gt;$-1$ TMMG not active&lt;br&gt;$-3$ no buffer magazine defined</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$P_{\text{MAGS}}$</td>
<td>INT</td>
<td>$P_{\text{MAGS}}[n]$&lt;br&gt;n-th number of the spindle / of the toolholder in the buffer&lt;br&gt;$&gt;0$ successful read access&lt;br&gt;$0$ n is outside the permitted range&lt;br&gt;$-1$ TMMG not active&lt;br&gt;$-3$ no buffer magazine defined&lt;br&gt;$n=1,...,\text{max. toolholder number}$</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$P_{\text{MAGNREL}}$</td>
<td>INT</td>
<td>$P_{\text{MAGNREL}}[n]$&lt;br&gt;Number of the buffer assigned to the spindle number / toolholder number n&lt;br&gt;$&gt;0$ successful read access&lt;br&gt;$0$ spindle location has no buffer location assigned&lt;br&gt;$-1$ TMMG not active&lt;br&gt;$-2$ n is not the number of a spindle location&lt;br&gt;$-3$ no buffer magazine defined&lt;br&gt;$n=1,...,\text{max. toolholder number}$</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$P_{\text{MAGREL}}$</td>
<td>INT</td>
<td>$P_{\text{MAGREL}}[n, m]$&lt;br&gt;m-th buffer number of the n-th spindle number / toolholder number&lt;br&gt;$&gt;0$ successful read access&lt;br&gt;$0$ m is outside the permitted range&lt;br&gt;$-1$ TMMG not active&lt;br&gt;$-2$ n is not the number of a spindle location&lt;br&gt;$-3$ no buffer magazine defined&lt;br&gt;$n=1,...,\text{max. toolholder number}$&lt;br&gt;$m=1,...,#P_{\text{MAGREL}}$</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| $P_{\text{MAGNH}}$ | INT   | $P_{\text{MAGNH}}$
Number of defined magazine location type hierarchies assigned to the channel.
$>0$ successful read access
$0$ no location type hierarchies are defined
$-1$ TMMG not active
OPI module = TT | R         | 6   | .  | 1 |
| $P_{\text{MAGNHLT}}$ | INT   | $P_{\text{MAGNHLT}}[n]$
Number of defined location types in the $n$th defined hierarchy
$>0$ successful read access
$0$ no location types are defined
$-1$ TMMG not active
OPI module = TT
$n=1,...,P_{\text{MAGNH}}$ | R         | 6   | .  | 1 |
| $P_{\text{MAGHLT}}$ | INT   | $P_{\text{MAGHLT}}[n,m]$
$m$-th location type of hierarchy $n$; $n=1,...,P_{\text{MAGNH}}$; $m=1,...,P_{\text{MAGNHLT}}$
$>0$ successful read access
$0$ no location types are defined
$-1$ TMMG not active
$-2$ hierarchy $n$ has no defined location types
OPI module = TT
$n=1,...,P_{\text{MAGNH}}$
$m=1,...,P_{\text{MAGNHLT}}$ | R         | 6   | .  | 1 |
| $P_{\text{MAGNA}}$ | INT   | $P_{\text{MAGNA}}$
Number of defined adapters assigned to the channel
$>0$ successful read access
$0$ no adapters defined
$-1$ TMMG or ‘Adapter’ function not active
OPI module = AD | R         | 6   | .  | 1 |
| $P_{\text{MAGA}}$ | INT   | $P_{\text{MAGA}}[i]$
i-th adapter number
$>0$ successful read access
$0$ no adapters defined
$-1$ TMMG or ‘Adapter’ function not active
OPI module = AD
$i=1,...,P_{\text{MAGNA}}$ | R         | 6   | .  | 1 |
| $P_{\text{MTHSDC}}$ | INT   | $P_{\text{MTHSDC}}$
Master toolholder no. or master spindle no. relative to that of the active tool
for which the next D offset selection is specified.
$>0$ successful read access
$0$ no master toolholder or master spindle available.
The next D offset works with T0.
$-1$ TMMG not available | R         | 6   | .  | 4 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\text{AC_MONMIN}$ | REAL    | $\text{AC_MONMIN}$  
Ratio between tool monitoring actual value and setpoint.  
Threshold for tool search strategy "load only tools with actual value greater than threshold" | R | WS | R | W | 6 | 1 |
| $\text{P_VDITCP}$ | INT     | $\text{P_VDITCP}[n]$  
Available parameters for magazine management on VDI interface  
n: Index 1–3 | R | W | | | 2 |
| $\text{A_DNO}$ | INT     | $\text{A_DNO}[i]$  
Read a D number defined by the PLC via VDI interface  
i: Index 1–9 for table location in D number table | R | | | | 4 |
| $\text{P_ATPG}$ | REAL    | $\text{P_ATPG}[n]$  
Current tool-related grinding data  
n: Parameter number 1–9 | R | W | | | 2 |
| $\text{P_TOOLENV}$ | STRING  | $\text{P_TOOLENV}[i]$  
Returns the name of the tool environment stored under (internal)  
index i. If i refers to a non-defined data block, the zero string is returned.  
If index i is invalid, in other words, if i is less than 1 or more than the maximum number of data blocks for tool environments ($\text{MN_MM_NUM_TOOLENV}$), an alarm is output.  
A maximum number of tool environments is configurable via MD $\text{MN_MM_NUM_TOOLENV}$.  
2nd dimension for TYPE_STRING is automatically $\text{MAXSTRINGLEN}$ | R | | | | 3 |
| $\text{P_TOOLENVN}$ | INT     | $\text{P_TOOLENVN}$  
Indicates the number of defined data blocks for describing tool environments. | R | | | | 6 |
| $\text{P_AP}$ | REAL    | $\text{P_AP}$  
Programmed angle for polar coordinates | R | | | | 6 |

#### 15.2.27 Programmed geometry axis values

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\text{P_AXN1}$ | AXIS  | $\text{P_AXN1}$  
Current address of the geometry axis - abscissa | R | | | | 3 |
| $\text{P_AXN2}$ | AXIS  | $\text{P_AXN2}$  
Current address of the geometry axis - ordinate | R | | | | 3 |
### 15.2 List of system variables

#### 15.2.28 G groups

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_AXN3$</td>
<td>AXIS</td>
<td>Current address of the geometry axis - applicate</td>
<td>R</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>$P_ACTGEOAX$</td>
<td>AXIS</td>
<td>$P_ACTGEOAX[1]$ Current geometry axis assignment, dependent on plane Returns the current geometry axis assignment programmed with geometry axis(1,X,2,Y,3,Z) Array index 1–3 for 1st to 3rd geometry axis n: Number of input 1 - ...</td>
<td>R</td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

#### 15.2.29 Programmed values

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_SEARCH$</td>
<td>BOOL</td>
<td>$P_SEARCH$ Block search is active if TRUE (1)</td>
<td>R</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$P_SEARCH1$</td>
<td>BOOL</td>
<td>$P_SEARCH1$ Block search with calculation is active if TRUE (1)</td>
<td>R</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$P_SEARCH2$</td>
<td>BOOL</td>
<td>$P_SEARCH2$ The last selected search type was block search without calculation, if TRUE (1)</td>
<td>R</td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $P\_SEARCHL | INT      | R1 = $P\_SEARCHL  
Returns the last selected search type:  
(coding analogous to PI service _N_FINDBL)  
0 : No block search  
1 : Block search without calculation  
2 : Block search with calculation on contour  
3 : Reserved  
4 : Block search with calculation at end of block position  
5 : Block search in extended program test | R         | 5 |
| $P\_SUBPAR  | BOOL     | $P\_SUBPAR[n]  
Query whether during subroutine call with parameter transfer  
parameter n was actually programmed (TRUE) or whether the system has  
set a default parameter (FALSE).  
n: Parameter number 1 to n corresponding to the definition in  
the PROC instruction | R         | 5 |
| $P\_CTABDEF | BOOL     | $P\_CTABDEF  
Definition of curve tables is active if TRUE (1) | R         | 4 |
| $P\_MC      | INT      | $P\_MC  
Status of modal subprogram call  
FALSE (0) -> Subprogram call not modal  
TRUE (1) -> Subprogram call modal | R         | 2 |
| $P\_REPINF  | INT      | $P\_REPINF  
Status info for repositioning with REPOS command  
(0) -> Repositioning with REPOS not possible for following  
reasons  
- Call is not executed in an ASUB  
- Call is executed in an ASUB, which was started in the reset  
state JOG mode  
- Call is executed in an ASUB, which was started in  
JOG mode  
(1) -> Repositioning with REPOS possible | R         | 4 |
| $P\_SIM     | BOOL     | $P\_SIM  
Simulation runs if TRUE (1) | R         | 2 |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_DRYRUN</td>
<td>BOOL</td>
<td>$P_DRYRUN</td>
<td>R</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dry run on if TRUE, else FALSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_OFFN</td>
<td>REAL</td>
<td>$P_OFFN</td>
<td>R</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Programmed offset contour normal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PI</td>
<td>REAL</td>
<td>$PI</td>
<td>R</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circle constant $\pi = 3.1415927</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_PROG_EVENT</td>
<td>INT</td>
<td>$P_PROG_EVENT can be used to query whether the program was implicitly activated by an event</td>
<td>R</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>configured by $MC_PROG_EVENT_MASK or by $MN_SEARCH_RUN_MODE. $P_PROG_EVENT returns an integer value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>between 0 and 5 with the following significance:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: explicit activation via NC Start or ASUB Start over the VDI or ASUB interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Implicit activation via event “Parts program start”</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Implicit activation via event “Parts program end”</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Implicit activation via event “Operator panel front reset”</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: Implicit activation via event “Booting”</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: Implicit activation subsequent to last action block display after block search</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_PROGPATH</td>
<td>STRING</td>
<td>$P_PROGPATH &lt;&lt; _N_MYSUB_SPF</td>
<td>R</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Call a subprogram from the current directory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: The current directory is /_N_WKS_DIR/ _N_SHAFT_DIR/.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The above call starts the subprogram /_N_WKS_DIR/ _N_SHAFT%_DIR/ _N_MYSUB_SPF.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_PROG</td>
<td>STRING</td>
<td>$P_PROG[0]</td>
<td>R</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns the program name of the program in program level 0 = main program name,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>in string variable NAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defines the program level from which the program name is to be read.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_STACK</td>
<td>INT</td>
<td>$P_STACK</td>
<td>R</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns the program level in which a parts program is active.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>progLevel = $P_STACK, writes in the integer variable the number of the current program level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>802S/C: Range of values = [0,5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>

**$P\_PATH**
- STRING
- **Identifier**
- **Type**
- **Description:** Reading the path name of the calling program.
- $P\_PATH[0]$ returns the directory of the current main program, for example, 
  
  `/N_WKS_DIR/N_WELLE_WPD`
- The variable is used, for example, to store a parts program generated with WRITE in the same directory as the calling program:
  
  ```
  PROC MYPRINTSUB
  DEF INT ERROR
  WRITE (ERROR, $P\_PATH[$P\_STACK - 1] << 
  "_N_LIST_MPF", "X10 Y20")
  ```
- If the subroutine was called from the shaft (WELLE) workpiece directory, a new file is generated in 
  `/N_WKS_DIR/N_WELLE_WPD/_N_LIST_MPF`.
- Defines the program level from which the program path is to be read
- 2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN

**$P\_ACTID**
- BOOL
- **Identifier**
- **Type**
- **Description:** Modal synchronized action with ID n active if TRUE
- n: 1–16

---

© Siemens AG, 2002. All rights reserved

SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
## 15.2.30 Channel states

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_STAT</td>
<td>INT</td>
<td>$AC_STAT&lt;br&gt;1: Invalid&lt;br&gt;0: Channel in reset mode&lt;br&gt;1: Channel interrupted&lt;br&gt;2: Channel active</td>
<td></td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AC_PROG</td>
<td>INT</td>
<td>$AC_PROG&lt;br&gt;1: Invalid&lt;br&gt;0: Program in reset mode&lt;br&gt;1: Program stopped&lt;br&gt;2: Program active&lt;br&gt;3: Program waiting&lt;br&gt;4: Program interrupted</td>
<td></td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AC_SYNA_MEM</td>
<td>INT</td>
<td>$AC_SYNA_MEM&lt;br&gt;Free memory for motion-synchronized actions&lt;br&gt;indicates how many elements of the memory assigned with $MC_MM_NUM_SYNC_ELEMENTS are still unassigned, readable from the parts program and the motion-synchronized actions</td>
<td></td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AC_IPO_BUF</td>
<td>INT</td>
<td>$AC_IPO_BUF&lt;br&gt;IPO buffer level, readable from the parts program and the motion-synchronized actions&lt;br&gt;The status is read from the parts program without feedforward stop while interpreting the block</td>
<td></td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AC_BLOCKTY</td>
<td>INT</td>
<td>$AC_BLOCKTYPE&lt;br&gt;Type of the current main run block.&lt;br&gt;0: Block is a programmed block (main block).&lt;br&gt;1: Block was not generated by the system as an intermediate block.</td>
<td></td>
<td>R</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$AC_TANEB</td>
<td>REAL</td>
<td>$AC_TANEB&lt;br&gt;Tangent ANgle at End of Block&lt;br&gt;The angle between the path tangent at the end point of the current block and the path tangent at the starting point of the following block.</td>
<td></td>
<td>R</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$AC_IW_STAT</td>
<td>INT</td>
<td>$AC_IW_STAT&lt;br&gt;Position information of the articulated joints (transformation-specific) for cartesian PTP travel</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$AC_IW_TU</td>
<td>INT</td>
<td>$AC_IW_TU&lt;br&gt;Position information of the axes (MCS) for cartesian PTP travel</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$AC_TRANS_SYS</td>
<td>INT</td>
<td>$AC_TRANS_SYS&lt;br&gt;Reference system for translation during the manual cartesian travel&lt;br&gt;0: axis-spec. manual travel active&lt;br&gt;1: cartesian manual travel in BCS&lt;br&gt;2: cartesian manual travel in WCS&lt;br&gt;3: cartesian manual travel in TCS</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_JOG_COO RD</td>
<td>INT</td>
<td>$AC_JOGCOORD Setting the coordinate system for manual travel 0: manual travel in WCS 1: manual travel in SZS</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$AC_ROT_SYS</td>
<td>INT</td>
<td>$AC_ROT_SYS Reference system for orientation during the manual cartesian travel 0: axis-spec. manual travel active 1: cartesian manual travel in BCS 2: cartesian manual travel in PCS 3: cartesian manual travel in TCS</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$A_PROBE</td>
<td>INT</td>
<td>$A_PROBE[1]: Status of first probe $A_PROBE[2]: Status of second probe 0 =&gt; not deflected 1 =&gt; deflected n: Number of probe</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AC_MEA</td>
<td>INT</td>
<td>$AC_MEA[n] Probe has been triggered if TRUE (1) n: Number of probe</td>
<td>R</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AC_TRAFO</td>
<td>INT</td>
<td>$AC_TRAFO Code number of the active transformation corresponding to machine data $MC_TRAFO_TYPE_n</td>
<td>RS</td>
<td>R</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$P_TRAFO</td>
<td>INT</td>
<td>$P_TRAFO Code number of the programmed transformation corresponding to machine data $MC_TRAFO_TYPE_n</td>
<td>R</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$AC_TRAFO_P AR</td>
<td>REAL</td>
<td>$AC_TRAFO_PAR[n] Parameter of the active transformation n: Number of the parameter</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$P_TRAFO_PA R</td>
<td>REAL</td>
<td>$P_TRAFO_PAR[n] Parameter of the programmed transformation n: Number of the parameter</td>
<td>R</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$AC_TRAFO_PA RSET</td>
<td>INT</td>
<td>$AC_TRAFO_PARSET Number of the active transformation record Variable is '0' if no transformer active</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$P_TRAFO_PA RSET</td>
<td>INT</td>
<td>$P_TRAFO_PARSET Number of the programmed transformation record Variable is '0' if no transformer active</td>
<td>R</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$AC_LIFTFAST</td>
<td>INT</td>
<td>Information about execution of liftfast. 0: Initial state. 1: Liftfast has been executed. At the start of the liftfast operation, the NC sets the value of the variable internally to &quot;1&quot;. The evaluating program (if available) must reset the variable to its initial setting ($AC_LIFTFAST=0) to enable a subsequent liftfast to be detected.</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_LIFTFAST</td>
<td>INT</td>
<td>$P_LIFTFAST Information about execution of liftfast. 0: Initial state. 1: Liftfast has been executed. At the start of the liftfast operation, the NC sets the value of the variable internally to &quot;1&quot;. The evaluating program (if available) must reset the variable to its initial setting to enable a subsequent liftfast to be detected. The reset is initiated by writing $AC_LIFTFAST!</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>$AC_ASUB</td>
<td>INT</td>
<td>$AC_ASUB Code number for the cause of the ASUB activation. The reasons are bit-coded and have the following significance: BIT0: Activation due to: User interrupt &quot;ASUB with Blsync&quot; Activation by: VDI signal, digital/analog interface Continuation by: user-selectable Reorg or Ret BIT1: Activation due to: User interrupt &quot;ASUB&quot; For program continuation with Repos, the position, after which the stop occurred is stored. Activation by: VDI signal, digital/analog interface Continuation by: user-selectable BIT2: Activation due to: User interrupt &quot;ASUB from Ready channel status&quot; Activation by: VDI signal, digital/analog interface Continuation by: user-selectable BIT3: Activation due to: User interrupt &quot;ASUB not READY in a manual mode and channel status&quot; Activation by: VDI signal, digital/analog interface Continuation by: user-selectable BIT4: Activation due to: User interrupt &quot;ASUB&quot; For program continuation with Repos, the actual position when the interrupt occurred is stored. Activation by: VDI signal, digital/analog interface Continuation by: user-selectable</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_ASUB</td>
<td>INT</td>
<td></td>
<td></td>
<td>RS</td>
<td>R</td>
<td>4</td>
</tr>
<tr>
<td>(continuation)</td>
<td></td>
<td><strong>BIT5:</strong> Activation due to: Cancelation of subprogram repetition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT6:</strong> Activation due to: Activation of decoding single block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI signal (+OPI)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT7:</strong> Activation due to: Activation of delete distance to go</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB Ret</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT8:</strong> Activation due to: Activation of axis synchronization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT9:</strong> Activation due to: Mode change</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB RET or RET (see MD.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT10:</strong> Activation due to: Program continuation with teach-in or after teach-in deactivation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI signal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB Ret</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT11:</strong> Activation due to: Overstore selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: Pt selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT12:</strong> Activation due to: Alarm with reaction of compensation block with Repos (COMPBLOCKWITHREORG)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: Internal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT13:</strong> Activation due to: Retraction movement on G33 and Stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: Internal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB Ret</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT14:</strong> Activation due to: Activation of dry run feedrate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BIT15:</strong> Activation due to: Deactivation of dry run feedrate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activation by: VDI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASUB</td>
<td>BIT16:</td>
<td>Activation due to: Activation of block suppression Activation by: VDI Continuation by: using system ASUB REPOS</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BIT17:</td>
<td>Activation due to: Deactivation of block suppression Activation by: VDI Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BIT18:</td>
<td>Activation due to: Set machine data active Activation by: Pi Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BIT19:</td>
<td>Activation due to: Set tool offset active Activation by: Pi &quot;_N_SETUDT&quot; Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BIT20:</td>
<td>Activation due to: System ASUB after search type SERUPRO has reached the search target. Activation by: Pi &quot;_N_FINDBL&quot; Parameter == 5 Continuation by: using system ASUB REPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_{ISTEST}$</td>
<td>BOOL</td>
<td>$P_{ISTEST}$ Check test mode in parts program TRUE = Program test active FALSE = Program test not active</td>
<td>R</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$P_{MMCA}$</td>
<td>STRING</td>
<td>$P_{MMCA}$ MMC task acknowledgment 2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN</td>
<td>R W</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$A_{PROTO}$</td>
<td>BOOL</td>
<td>$A_{PROTO}$ Activate / disable Logging function for the first user</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$A_{PROTOC}$</td>
<td>BOOL</td>
<td>$A_{PROTOC}$ Activate / disable Logging function for a user 0–9, USER</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
### 15.2.31 Synchronized actions

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC_FIFO1}$</td>
<td>REAL</td>
<td>$\text{AC_FIFO1}[n]$ \nFIFO for motion-synchronized actions and cyclic measurements \nn: Parameter number 0 - max. FIFO element \nSpecial meaning: \nn=0: On write accesses with index 0, a new value is stored in the FIFO, \nOn read accesses with index 0, the oldest element is read and deleted from the FIFO \nn=1: Read access to oldest element \nn=2: Read access to latest element \nn=3: Sum of all the elements located in the FIFO when in MD $\text{MC_MM_MODE_FIFO}$, bit0 is set \nn=4: Read access to current number of FIFO elements \nn=5-m: Read access to individual FIFO elements \n5 is the oldest element \n6 is the second-oldest, etc.</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{AC_FIFO2}$</td>
<td>REAL</td>
<td>$\text{AC_FIFO2}[n]$ \nFIFO for motion-synchronized actions and cyclic measurements \nn: Parameter number 0 - max. FIFO element \nSpecial meaning: \nn=0: On write accesses with index 0, a new value is stored in the FIFO, \nOn read accesses with index 0, the oldest element is read and deleted from the FIFO \nn=1: Read access to oldest element \nn=2: Read access to latest element \nn=3: Sum of all the elements located in the FIFO when in MD $\text{MC_MM_MODE_FIFO}$, bit0 is set \nn=4: Read access to current number of FIFO elements \nn=5-m: Read access to individual FIFO elements \n5 is the oldest element \n6 is the second-oldest, etc.</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\text{AC_FIFO10}$</td>
<td>REAL</td>
<td>$\text{AC_FIFO10}[n]$ \nas $\text{AC_FIFO01}$ ...</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
### 15.2.32 I/Os

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_IN$</td>
<td>BOOL</td>
<td>$A_IN[n]$ Digital input NC</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Number of input 1 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. input number results from MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$MN_FASTIO_DIG_NUM_INPUTS$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_OUT$</td>
<td>BOOL</td>
<td>$A_OUT[n]$ Digital output NC</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Number of output 1 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. input number results from MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$MN_FASTIO_DIG_NUM_OUTPUTS$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_INA$</td>
<td>REAL</td>
<td>$A_INA[n]$ Analog input NC</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Number of input 1 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. input number results from MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$MN_FASTIO_ANA_NUM_INPUTS$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_OUTA$</td>
<td>REAL</td>
<td>$A_OUTA[n]$ Analog output NC When writing, the value does not become active until the next IPO cycle at which point it is read back.</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Number of output 1 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. input number results from MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$MN_FASTIO_ANA_NUM_OUTPUTS$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_INCO$</td>
<td>BOOL</td>
<td>$A_INCO[n]$ Comparator input</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Number of output 1 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The max. input number results from the MD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 15.2.33 Reading and writing PLC variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_DBB$</td>
<td>INT</td>
<td>$A_DBB[n]$ Read/write data byte (8 bits) from/to PLC</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Position offset within I/O area 0 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_DBW$</td>
<td>INT</td>
<td>$A_DBW[n]$ Read/write data word (16 bits) from/to PLC</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Position offset within I/O area 0 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_DBD$</td>
<td>INT</td>
<td>$A_DBD[n]$ Read/write double data word (32 bits) from/to PLC</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Position offset within I/O area 0 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_DBR$</td>
<td>REAL</td>
<td>$A_DBR[n]$ Read/write Real data (32 bits) from/to PLC</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Position offset within I/O area 0 - ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 15.2.34 NCU link

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_DLB</td>
<td>INT</td>
<td>$A_DLB[n]$ Read/write data byte (8 bits) from/to NCU link n: Position offset within the link memory area 0 - ...</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$A_DLW</td>
<td>INT</td>
<td>$A_DLW[n]$ Read/write data word (16 bits) from/to NCU link n: Position offset within the link memory area 0 - ... synchronized with main run</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$A_DLD</td>
<td>INT</td>
<td>$A_DLD[n]$ Read/write data double word (32 bits) from/to NCU link n: Position offset within the link memory area 0 - ... synchronized with main run</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$A_DLR</td>
<td>REAL</td>
<td>$A_DLR[n]$ Read/write Real data (32 bits) from/to NCU link n: Position offset within the link memory area 0 - ... synchronized with main run</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$A_LINK_TRANS_RATE</td>
<td>INT</td>
<td>$A_LINK_TRANS_RATE Number of bytes that can still be transferred via NCU link Communication in the current IPO cycle.</td>
<td>R</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 15.2.35 Direct PLC I/O

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_PBB_IN</td>
<td>INT</td>
<td>$A_PBB_IN[n]$ Read data byte (8 bits) directly from PLC I/O n: Position offset within PLC input area 0 - ...</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$A_PBW_IN</td>
<td>INT</td>
<td>$A_PBW_IN[n]$ Read data word (16 bits) directly from PLC I/O n: Position offset within PLC input area 0 - ...</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$A_PBD_IN</td>
<td>INT</td>
<td>$A_PBD_IN[n]$ Read data double word (32 bits) directly from PLC I/O n: Position offset within PLC input area 0 - ...</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$A_PBR_IN</td>
<td>REAL</td>
<td>$A_PBR_IN[n]$ Read Real data (32 bits) directly from PLC I/O n: Position offset within PLC input area 0 - ...</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_PBB_OUT</td>
<td>INT</td>
<td>Write data byte (8 bits) directly to PLC I/O; n: Position offset within PLC output area 0 - ...; synchronized with main run</td>
<td>RS W R W 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_PBW_OUT</td>
<td>INT</td>
<td>Write data word (16 bits) directly to PLC I/O; n: Position offset within PLC output area 0 - ...; synchronized with main run</td>
<td>RS W R W 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_PBD_OUT</td>
<td>INT</td>
<td>Write data double word (32 bits) directly to PLC I/O; n: Position offset within PLC output area 0 - ...; synchronized with main run</td>
<td>RS W R W 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_PBR_OUT</td>
<td>REAL</td>
<td>Write Real data (32 bits) directly to PLC I/O; n: Position offset within PLC output area 0 - ...; synchronized with main run</td>
<td>RS W R W 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_IN</td>
<td>BOOL</td>
<td>Signal from PLC to Cycle; reserved for SIEMENS applications; n: Number of input 1 - ...</td>
<td>RS R 6 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_OUT</td>
<td>BOOL</td>
<td>Signal from Cycle to PLC; reserved for SIEMENS applications; n: Number of output 1 - ...</td>
<td>RS W R W 6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 15.2.36 Tool management

These system variables have value –1 if no tool management command is active at time of reading.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_TC_CMDT</td>
<td>INT</td>
<td>Trigger variable: $AC_TC_CMDT (CoMmandTrigger) then always takes on the value 1 for an interpolation cycle when a new magazine management command is output to the PLC.</td>
<td>RS R 6 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_TC_ACKT</td>
<td>INT</td>
<td>Trigger variable: $AC_TC_ACKT (ACKnowledgeTrigger) then always takes on the value 1 for an interpolation cycle when the PLC acknowledges a tool management command.</td>
<td>RS R 6 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC_TC_CMPD}$</td>
<td>INT</td>
<td>Counter variable: $\text{AC_TC_CMPD}$ (ComMmandCounter) is incremented by 1 for each tool management command output to the PLC. synchronized with main run.</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{AC_TC_ACKC}$</td>
<td>INT</td>
<td>Counter variable: $\text{AC_TC_ACKC}$ (ACKnowledgeCounter) on acknowledging a tool management command is incremented by 1 via the PLC. synchronized with main run.</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{AC_TC_FCT}$</td>
<td>INT</td>
<td>Command number. Specifies which action is desired. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_STATUS}$</td>
<td>INT</td>
<td>Status enjoyed by the command to read via $\text{AC_TC_FCT}$. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_THNO}$</td>
<td>INT</td>
<td>Number of the toolholder (spec. the spindle no.) where the new tool is to be changed. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_TNO}$</td>
<td>INT</td>
<td>NCK-internal T number of new tool (to be changed). 0: There is no new tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_MMYN}$</td>
<td>INT</td>
<td>Owner magazine number of old tool (to be changed). 0: There is no new tool, or the new tool (if $\text{AC_TC_TNO} &gt; 0$) is not loaded (manual tool). -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_LMYN}$</td>
<td>INT</td>
<td>Owner location number of old tool (to be changed). 0: There is no new tool, or the new tool (if $\text{AC_TC_TNO} &gt; 0$) is not loaded (manual tool). -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_MFN}$</td>
<td>INT</td>
<td>Source magazine number of new tool. 0: There is no new tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{AC_TC_LFN}$</td>
<td>INT</td>
<td>Source location number of new tool. 0: There is no new tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr</td>
<td>Syn</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td>----------</td>
<td>-----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$AC_TC_MTN</td>
<td>INT</td>
<td>$AC_TC_MTN Target magazine number of new tool. 0: There is no new tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_TC_LTN</td>
<td>INT</td>
<td>$AC_TC_LTN Target location number of new tool. 0: There is no new tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_TC_MFO</td>
<td>INT</td>
<td>$AC_TC_MFO Source magazine number of old tool (to be changed). 0: There is no old tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_TC_LFO</td>
<td>INT</td>
<td>$AC_TC_LFO Source location number of old tool (to be changed). 0: There is no old tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_TC_MTO</td>
<td>INT</td>
<td>$AC_TC_MTO Target magazine number of old tool (to be changed). 0: There is no old tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_TC_LTO</td>
<td>INT</td>
<td>$AC_TC_LTO Target location number of old tool (to be changed). 0: There is no old tool. -1: No tool management command is active at the time of reading.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.37 Timers

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Syn</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\$A\_YEAR    | INT     | \$A\_YEAR  
System time, year | RS | R | 3 |
| $\$A\_MONTH   | INT     | \$A\_MONTH  
System time, month | RS | R | 3 |
| $\$A\_DAY     | INT     | \$A\_DAY  
System time, day | RS | R | 3 |
| $\$A\_HOUR    | INT     | \$A\_HOUR  
System time, hour | RS | R | 3 |
| $\$A\_MINUTE  | INT     | \$A\_MINUTE  
System time, minute | RS | R | 3 |
| $\$A\_SECOND  | INT     | \$A\_SECOND  
System time, second | RS | R | 3 |
| $\$A\_MSECOND | INT     | \$A\_MSECOND  
System time, millisecond | RS | R | 3 |
| $\$AC\_TIME   | REAL    | \$AC\_TIME  
Time from the beginning of block in seconds 
This variable can only be accessed from synchronized actions | RS | R | 2 |
| $\$AC\_TIMEC  | REAL    | \$AC\_TIMEC  
Time from the beginning of block in IPO clock cycles 
This variable can only be accessed from synchronized actions | RS | R | 3 |
| $\$AC\_TIMER  | REAL    | \$AC\_TIMER[n]  
Timer - unit in seconds 
Time is counted internally in multiples of the interpolation cycle; 
Counting for the time variable is started by assigning the value \$AC\_TIMER[n]=<starting value> 
To stop the counter variable, assign a negative value: \$AC\_TIMER[n]=-1 
The current time can be read while the counter is active or stopped. Stopping the time variable, by assigning - 1 stops the last current time value which can then be read 
The dimension is defined in MD $MC\_MM\_NUM\_AC\_TIMER." | RS | WS | R | W | + | 4 |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AC_PRTIME_M    | REAL   | $AC_PRTIME_M      “ProgramRunTIME-Main”  
Set (initialize) the accumulated program runtime (main time)                          | W         |      |   |   |
| $AC_PRTIME_A    | REAL   | $AC_PRTIME_A      “ProgramRunTIME-Auxiliary”  
Set (initialize) the accumulated program runtime (auxiliary time)                           | W         |      |   |   |
| $AC_PRTIME_M    | REAL   | $AC_PRTIME_M_INC  “ProgramRunTIME-Main-INCrement”  
Increment the accumulated program runtime (main time)                                    | W         |      |   |   |
| $AC_PRTIME_A    | REAL   | $AC_PRTIME_A_INC  “ProgramRunTIME-Auxiliary-INCrement”  
Increment the accumulated program runtime (auxiliary time)                              | W         |      |   |   |

15.2.38 Path movement

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AC_PATHN       | REAL   | $AC_PATHN
Normalized path parameter
value between 0=start of block and 1=end of block
This variable can only be accessed from synchronized actions                                          | RS        |      | R | 2 |
| $AC_DTBW        | REAL   | $AC_DTBW
Geometric distance from start of block in workpiece coordinate system
The programmed position is decisive for computing the distance; if the axis is a coupling axis, the position part that results from axis coupling is not considered here.
This variable can only be accessed from synchronized actions                                          | RS        |      | R | 2 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\$AC\_REPOS\_PATH\_MODE | INT      | $\$AC\_REPOS\_PATH\_MODE
RePOS mode type

0  = not defined.
1  = RMB
Repositioning approach to start of interrupted block
2  = RMI
Repositioning approach to interruption point of interrupted block
3  = RME
Repositioning approach to end of interrupted block
4  = RMN
Repositioning approach to geometrically nearest point of interrupted block

The variable is defined just as REPOS is being executed, or if a new REPOS mode has been specified via VDI. | RS     | R   | 6 | - | 4 |
| $\$AC\_DTBB | REAL     | $\$AC\_DTBB
Geometric distance from start of block in basic coordinate system

The programmed position is decisive for computing the distance; if the axis is a coupling axis, the position part that results from axis coupling is not considered here.
This variable can only be accessed from synchronized actions | RS     | R   | 2 |
| $\$AC\_DTEW | REAL     | $\$AC\_DTEW
Geometric distance from end of block in workpiece coordinate system

The programmed position is decisive for computing the distance; if the axis is a coupling axis, the position part that results from axis coupling is not considered here.
This variable can only be accessed from synchronized actions | RS     | R   | 2 |
| $\$AC\_DTEB | REAL     | $\$AC\_DTEB
Geometric distance from end of block in basic coordinate system

The programmed position is decisive for computing the distance; if the axis is a coupling axis, the position part that results from axis coupling is not considered here.
This variable can only be accessed from synchronized actions | RS     | R   | 2 |
| $\$AC\_PLTBB | REAL     | $\$AC\_PLTBB
Path distance from start of block in basic coordinate system

This variable can only be accessed from synchronized actions | RS     | R   | 3 |
| $\$AC\_PLTEB | REAL     | $\$AC\_PLTEB
Path distance from end of block in basic coordinate system

This variable can only be accessed from synchronized actions | RS     | R   | 3 |
### 15.2 List of system variables

#### Speeds/accelerations

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\$AC\_DEL T | REAL   | $\$AC\_DEL T  
Stored distance-to-go path in the workpiece coordinate system subsequent to deletion of the residual distance during synchronized motion actions | RW        | 3    |   |   |
| $\$P\_APDV | BOOL   | $\$P\_APDV  
Returns True if the position values readable with $\$P\_APR[X] and $\$P\_AEP[X] (approach and depart starting point and contour point when smoothing) are valid. | R         | 4    |   |   |
| $\$P\_F    | REAL   | $\$P\_F    
Path feed F last programmed                                                                                     | R         | 2    |   |   |
| $\$AC\_F   | REAL   | $\$AC\_F   
Programmed path feed F                                                                                         | R         | 6    | 3 |   |
| $\$AC\_OVR | REAL   | $\$AC\_OVR:  
Path override for synchronized actions  
Multiplicative override component, works in addition to operation OV, programmed OV and transformation OV.  
However, the overall factor remains limited to the maximum value defined by machine data $\$MN\_OVR\_FACTOR\_LIMIT\_BIN and $\$MN\_OVR\_FACTOR\_FEEDRATE[31].  
If a value of < 0.0 is entered, 0 is assumed and alarm 14756 reported.  
Must be rewritten in every interpolator cycle, otherwise the value is 100%.  
This variable can only be accessed from synchronized actions | R W      | 2    |   |   |
# 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AC_VC    | REAL  | $AC_VC  
Additive path feed compensation for synchronized actions
The compensation value does not work with G0, G33, G331, G332 and G63.
It must be rewritten in every interpolator cycle, otherwise the value is 0.
With an override of 0, the compensation value has no effect, otherwise the override has no impact on the compensation value.
The compensation value cannot make the total feedrate negative.
The upper value is limited such that the maximum axis velocities and accelerations are not exceeded.
The computation with different feedrate components is not affected by $AC_VC.
The override defined by machine data $MN_OVR_FACTOR_LIMIT_BIN, $MN_OVR_FACTOR_FEEDRATE[30], $MN_OVR_FACTOR_AX_SPEED[30] and $MN_OVR_FACTOR_SPIND_SPEED Override values cannot be exceeded. The additive feedrate override is limited such that the resulting feedrate does not exceed the maximum override value of the programmed feedrate.
This variable can only be accessed from synchronized actions | R | W | 2 |
| $AC_PATHACC | REAL  | $AC_PATHACC  
Specification of an increased path acceleration for override changes and Stop/Start events.
$AC_PATHACC is only considered when the value is greater than the prepared acceleration limitation.
The value 0 deselects the function.
Values that lead to machine axis accelerations that are twice as high as that parameterized in $MA_MAX_AX_ACCEL[;]. are accordingly restricted internally. | RS | WS | R | W | 6 | 3 |
| $AC_PATHJERK | REAL | $AC_PATHJERK  
Specification of an increased path jerk for override changes and Stop/Start events.
$AC_PATHJERK is only considered when the value is greater than the prepared jerk limitation.
The value 0 deselects the function. | RS | WS | R | W | 6 | 3 |
| $AC_VACTB | REAL  | $AC_VACTB  
Path velocity in the base coordinate system
This variable can only be accessed from synchronized actions | RS | R | 2 |
| $AC_VACTW | REAL  | $AC_VACTW  
Path velocity in workpiece coordinate system
This variable can only be accessed from synchronized actions | RS | R | 2 |
### 15.2.40 Spindles

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_S$</td>
<td>REAL</td>
<td>$P_S[n]$ Last programmed spindle speed</td>
<td>R</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_S$</td>
<td>REAL</td>
<td>$AA_S[n]$ Spindle actual speed. The sign corresponds to the direction of rotation.</td>
<td>RS R</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_{CONSTCUT}_S$</td>
<td>REAL</td>
<td>$P_{CONSTCUT_S}[n]$ Last programmed constant cutting speed.</td>
<td>R</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_{CONSTCU}_T_S$</td>
<td>REAL</td>
<td>$AC_{CONSTCUT_S}[n]$ Current constant cutting speed.</td>
<td>RS R</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_{SEARCH_S}$</td>
<td>REAL</td>
<td>$P_{SEARCH_S}[n]$ The last programmed spindle speed or cutting rate picked up during search mode</td>
<td>R</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_{SDIR}$</td>
<td>INT</td>
<td>$P_{SDIR}[n]$ Last direction of spindle rotation to be programmed.</td>
<td>R</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Clockwise spindle rotation,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: Counterclockwise spindle rotation,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: Spindle stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_{SDIR}$</td>
<td>INT</td>
<td>$AC_{SDIR}[n]$ Current direction of spindle rotation</td>
<td>RS R</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Clockwise spindle rotation,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: Counterclockwise spindle rotation,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: Spindle stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_{SEARCH_SDIR}$</td>
<td>INT</td>
<td>$P_{SEARCH_SDIR}[n]$ The last programmed spindle programming picked up during search mode:</td>
<td>R</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: M3 Clockwise spindle rotation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: M4 Counterclockwise spindle rotation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: M5 Spindle stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-19: M19, SPOS, SPOSA spindle positioning, position and approach mode are read from SEARCH variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>70: M70 Switch over to axis mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-5: No direction of spindle rotation programmed, no output.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Sync</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td>----------</td>
<td>------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$P_SMODE</td>
<td>INT</td>
<td>$P_SMODE[n]$ Last programmed spindle mode: 0: No spindle in the channel or spindle is active in another channel or is being used by the PLC (FC18) or by synchronized actions. 1: Speed control mode 2: Positioning mode 3: synchronized mode 4: Axis mode n: Spindle number 0 ... max. spindle number</td>
<td>R</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>$AC_SMODE</td>
<td>INT</td>
<td>$AC_SMODE[n]$ Spindle mode currently active 0: No spindle in the channel 1: Speed control mode 2: Positioning mode 3: synchronized mode 4: Axis mode n: Spindle number 0 ... max. spindle number</td>
<td>RS R</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>$P_SGEAR</td>
<td>INT</td>
<td>$P_SGEAR[n]$ Spindle gear stage last programmed or requested during M40 by S programming 1: 1st gear stage requested .... 5: 5th gear stage requested n: Spindle number 0 ... max. spindle number</td>
<td>R</td>
<td></td>
<td></td>
<td>6 1</td>
</tr>
<tr>
<td>$AC_SGEAR</td>
<td>INT</td>
<td>$AC_SGEAR[n]$ Active spindle gear stage 1: 1st gear stage is active .... 5: 5th gear stage is active n: Spindle number 0 ... max. spindle number</td>
<td>RS R</td>
<td></td>
<td></td>
<td>6 1</td>
</tr>
<tr>
<td>$P_SAUTOGEAR</td>
<td>INT</td>
<td>$P_SAUTOGEAR[n]$ Automatic gear stage change (M40) is programmed. 0: Gear stages are requested by M41..M45 1: The gear stage is calculated and requested (M40 Automatic gear stage change is active) to fit the programmed speed (S) n: Spindle number 0 ... max. spindle number</td>
<td>R</td>
<td></td>
<td></td>
<td>6 1</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_SEARCH_SGEAR</td>
<td>INT</td>
<td>The last programmed gear stages M function picked up during search mode.</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>40: M40 Automatic gear stage change</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>41: M41 1st gear stage requested</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>45: M45 5th gear stage requested</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_SEARCH_SPOS</td>
<td>REAL</td>
<td>The last spindle position or travel path programmed by M19, SPOS or SPOSA picked up during search mode.</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Position: 0...359.999, when the value in MD 30330 MODULO_RANGE is 360.0 degrees</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Path: -100000000 ... 100000000 degrees. The sign indicates the traversing direction.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_SEARCH_SPOSMODE</td>
<td>INT</td>
<td>The last position approach mode programmed by M19, SPOS or SPOSA picked up during search mode.</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: DC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: IC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: DC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: ACP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5: ACN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ... max. spindle number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_NUM_SPINDLES</td>
<td>INT</td>
<td>Calculates the maximum number of spindles in the channel.</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No spindle in the channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1..n: Number of spindles in the channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_MSNUM</td>
<td>INT</td>
<td>Returns the number of the master spindle.</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No spindle in the channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1..n: Number of the master spindle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_MSNUM</td>
<td>INT</td>
<td>Returns the number of the current master spindle.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No spindle exists</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1..n: Number of the master spindle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_MTHNUM</td>
<td>INT</td>
<td>Returns the number of the master tool carrier.</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No master tool carrier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1..n: Number of master tool carrier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_MTHNUM</td>
<td>INT</td>
<td>$AC_MTHNUM - only useful with active magazine management&lt;br&gt;Returns the number of the current master tool carrier:&lt;br&gt;0: No master tool carrier&lt;br&gt;1..n: Number of master tool carrier</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$P_GWPS</td>
<td>BOOL</td>
<td>$P_GWPS[n] Constant grinding wheel surface speed on if TRUE&lt;br&gt;n: Spindle number</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 15.2.41 Polynomial values for synchronized actions

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_FCT1LL</td>
<td>REAL</td>
<td>$AC_FCT1LL Lower limit value for evaluation function FCTDEF 1</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT2LL</td>
<td>REAL</td>
<td>$AC_FCT2LL Lower limit value for evaluation function FCTDEF 2</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT3LL</td>
<td>REAL</td>
<td>$AC_FCT3LL Lower limit value for evaluation function FCTDEF 3</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT1UL</td>
<td>REAL</td>
<td>$AC_FCT1UL Upper limit value for evaluation function FCTDEF 1</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT2UL</td>
<td>REAL</td>
<td>$AC_FCT2UL Upper limit value for evaluation function FCTDEF 2</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT3UL</td>
<td>REAL</td>
<td>$AC_FCT3UL Upper limit value for evaluation function FCTDEF 3</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT1C[n]</td>
<td>REAL</td>
<td>Polynomial coefficient a0–a3 for evaluation function FCTDEF 1&lt;br&gt;n: Degree of coefficient 0–3</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT2C[n]</td>
<td>REAL</td>
<td>Polynomial coefficient a0–a3 for evaluation function FCTDEF 2&lt;br&gt;n: Degree of coefficient 0–3</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCT3C[n]</td>
<td>REAL</td>
<td>Polynomial coefficient a0–a3 for evaluation function FCTDEF 3&lt;br&gt;n: Degree of coefficient 0–3</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCTLL[n]</td>
<td>REAL</td>
<td>Lower limit of polynomial for synchronized actions (SYNFCT)&lt;br&gt;n: Number of polynomial, limited by machine data</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AC_FCTUL[n]</td>
<td>REAL</td>
<td>Upper limit of polynomial for synchronized actions (SYNFCT)&lt;br&gt;n: Number of polynomial, limited by machine data</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_FCT0</td>
<td>REAL</td>
<td>$AC_FCT0[n] a0 coefficient of polynomial for synchronized actions (SYNFCT) n: Number of polynomial, limited by machine data</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W+4</td>
</tr>
<tr>
<td>$AC_FCT1</td>
<td>REAL</td>
<td>$AC_FCT1[n] a1 coefficient of polynomial for synchronized actions (SYNFCT) n: Number of polynomial, limited by machine data</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W+4</td>
</tr>
<tr>
<td>$AC_FCT2</td>
<td>REAL</td>
<td>$AC_FCT2[n] a2 coefficient of polynomial for synchronized actions (SYNFCT) n: Number of polynomial, limited by machine data</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W+4</td>
</tr>
<tr>
<td>$AC_FCT3</td>
<td>REAL</td>
<td>$AC_FCT3[n] a3 coefficient of polynomial for synchronized actions (SYNFCT) n: Number of polynomial, limited by machine data</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W+4</td>
</tr>
</tbody>
</table>
### 15.2.42 Channel states

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{SAC_ALARM_STAT}$</td>
<td>INT</td>
<td>$\text{SAC_ALARM_STAT}$ (Selected) alarm reactions for synchronized actions (SYNFCT)</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\text{SAN_ESR_TRIGGER}$</td>
<td>BOOL</td>
<td>$\text{SAN_ESR_TRIGGER} = 1$ Trigger &quot;Extended stop and retract&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SAN_BUS_FAIL_TRIGGER}$</td>
<td>BOOL</td>
<td>$\text{SAN_BUS_FAIL_TRIGGER} = 1$ Simulation of a drive bus failure for test purposes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SAC_ESR_TRIGGER}$</td>
<td>BOOL</td>
<td>$\text{SAC_ESR_TRIGGER} = 1$ Triggering &quot;NC controlled ESR&quot;</td>
<td>R W</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{SAC_OPERATING_TIME}$</td>
<td>REAL</td>
<td>IF $\text{SAC_OPERATING_TIME} &lt; 12000$ GOTOB STARTMARK Total execution time ( in seconds ) of NC programs in automatic mode</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAC_CYCLE_TIME}$</td>
<td>REAL</td>
<td>IF $\text{SAC_CYCLE_TIME} &gt; 2400$ GOTOF ALARM01 Execution time of the selected NC program ( in seconds )</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAC_CUTTING_TIME}$</td>
<td>REAL</td>
<td>IF $\text{SAC_CUTTING_TIME} &gt; 6000$ GOTOF ACT_M06 Tool operation time ( in seconds )</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAC_REQUIRED_PARTS}$</td>
<td>REAL</td>
<td>$\text{SAC_REQUIRED_PARTS} = \text{ACTUAL_LOS}$ Definition of number of parts required, e.g. for definition of a batch size, daily production target, etc.</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAC_TOTAL_PARTS}$</td>
<td>REAL</td>
<td>IF $\text{SAC_TOTAL_PARTS} &gt; \text{SERVICE_COUNT}$ GOTOF MARK_END Total number of all parts produced</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAC_ACTUAL_PARTS}$</td>
<td>REAL</td>
<td>IF $\text{SAC_ACTUAL_PARTS} == 0$ GOTOF NEW_RUN Actual number of parts produced For $\text{SAC_ACTUAL_PARTS} == \text{SAC_REQUIRED_PARTS}$, $\text{SAC_ACTUAL_PARTS} = 0$ automatically.</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\text{SAC_SPECIAL_PARTS}$</td>
<td>REAL</td>
<td>$\text{SAC_SPECIAL_PARTS} = \text{R20}$ Number of parts counted according to a user strategy. without internal impact.</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC}_\text{G0MODE}$</td>
<td>INT</td>
<td>$\text{AC}_\text{G0MODE}$</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: G0 not active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: G0 and linear interpolation active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: G0 and nonlinear interpolation active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The response of the path axes at G0 is dependent on machine data $\text{MC}<em>\text{G0_LINEAR_MODE}$ (Siemens mode) or $\text{MC}</em>\text{EXTERN}_\text{G0_LINEAR_MODE}$ (ISO mode): with linear interpolation, the path axes traverse together, with non-linear interpolation, the path axes traverse as positioning axes.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 15.2.43 Measurement

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC}_\text{MEAS_SEMA}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_SEMA} = 1$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Assigning the measuring interface.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AC}_\text{MEAS_LATCH}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_LATCH}[0] = 1$</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Writing the actual axis values to the 1st measuring point.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: 1st measuring point,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.. ,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: 4th measuring point</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AC}_\text{MEAS_P1_COORD}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_P1_COORD} =$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: WCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: BCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: MCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinate system of the 1st measuring point.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AC}_\text{MEAS_P2_COORD}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_P2_COORD} =$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: WCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: BCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: MCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinate system of the 2nd measuring point.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AC}_\text{MEAS_P3_COORD}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_P3_COORD} =$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: WCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: BCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: MCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinate system of the 3rd measuring point.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AC}_\text{MEAS_P4_COORD}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_P4_COORD} =$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: WCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: BCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: MCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinate system of the 4th measuring point.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{AC}_\text{MEAS_SET_COORD}$</td>
<td>INT</td>
<td>$\text{AC}_\text{MEAS_SET_COORD} =$</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: WCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: BCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: MCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinate system of the setpoint.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Sync</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>------------------------------------------------</td>
<td>-----------</td>
<td>------</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| $AC_MEAS_WP_SETANGLE | REAL    | $AC_MEAS_WP_SETANGLE = 0.0
Setpoint angle of the part position for part gauging. | R         | W    |   |   |
| $AC_MEAS_CORNER_SETANGLE | REAL    | $AC_MEAS_CORNER_SETANGLE = 90.0
Setpoint cutting angle of the corner for part gauging. | R         | W    |   |   |
| $AC_MEAS_DIR_APPROACH | INT     | $AC_MEAS_DIR_APPROACH =
0: +x
1: -x
2: +y
3: -y
4: +z
5: -z
Direction of approach to the part. | R         | W    |   |   |
| $AC_MEAS_ACT_PLANE | INT     | $AC_MEAS_ACT_PLANE =
0: G17
1: G18
2: G19
Setting the plane for calculation and measuring. | R         | W    |   |   |
| $AC_MEAS_FINE_TRANS | INT     | $AC_MEAS_FINE_TRANS =
0: Offset in Trans
1: Offset in Fine Trans
Setting the fine offset for calculation and measuring. | R         | W    |   |   |
| $AC_MEAS_FRAME_SELECT | INT     | $AC_MEAS_FRAME_SELECT =
0: $P_SETFRAME
10..25: $P_CHBFRAME[0..15]
50..65: $P_NCBFRAME[0..15]
100..199: $P_IFRAME
1010..1025: $P_CHBFRAME[0..15], with active G500
1050..1065: $P_NCBFRAME[0..15], with active G500
2000: $P_SETFRAME
2010..2025: $P_CHBFRAME[0..15]
2050..2065: $P_NCBFRAME[0..15]
2100..2199: $P_UIFR[0..99]
3010..3025: $P_CHBFRAME[0..15], with active G500
3050..3065: $P_NCBFRAME[0..15], with active G500
Selecting the frames for part gauging. | R         | W    |   |   |
| $AC_MEAS_CHSFR | INT     | $AC_MEAS_CHSFR = “B1001”
System frame bit mask in accordance with $MC_MM_SYSTEM_FRAME_MASK | R         | W    |   |   |
| $AC_MEAS_NCBFR | INT     | $AC_MEAS_NCBFR = “B1”
Global basic frame mask to set up the new frame. | R         | W    |   |   |
| $AC_MEAS_CHBF | INT     | $AC_MEAS_CHBF = “B1”
Channel basic frame mask to set up the new frame. | R         | W    |   |   |
| $AC_MEAS_UIFR | INT     | $AC_MEAS_UIFR = 1
Adjustable data management frame to set up the new frame. | R         | W    |   |   |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC_MEAS_PF RAME</td>
<td>INT</td>
<td>$AC_MEAS_PFRAME = 1 Programmable frame not included.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$AC_MEAS_T_ NUMBER</td>
<td>INT</td>
<td>$AC_MEAS_T_NUMBER = 1 Selecting the tool for calculation and measuring.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$AC_MEAS_TO OL_MASK</td>
<td>INT</td>
<td>$AC_MEAS_TOOL_MASK = 'B1' Setting the tool for calculation and measuring.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$AC_MEAS_D_ NUMBER</td>
<td>INT</td>
<td>$AC_MEAS_D_NUMBER = 1 Selecting the cutting edge for calculation and measuring.</td>
<td>R</td>
<td>W</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AC\_MEAS\_VALID$ | INT    | $AC\_MEAS\_VALID = 0$  
Validity bits of the measuring variables. The value should be set to 0 prior to all measuring processes. The individual bits are set implicitly when writing to the corresponding variables.  
Bit 0: $AA\_MEAS\_POINT1[\text{axis}]$  
Bit 1: $AA\_MEAS\_POINT2[\text{axis}]$  
Bit 2: $AA\_MEAS\_POINT3[\text{axis}]$  
Bit 3: $AA\_MEAS\_POINT4[\text{axis}]$  
Bit 4: $AA\_MEAS\_SETPOINT[\text{axis}]$  
Bit 5: $AC\_MEAS\_WP\_SETANGLE$  
Bit 6: $AC\_MEAS\_CORNER\_SETANGLE$  
Bit 7: $AC\_MEAS\_T\_NUMBER$  
Bit 8: $AC\_MEAS\_D\_NUMBER$  
Bit 9: $AC\_MEAS\_DIR\_APPROACH$  
Bit 10: $AC\_MEAS\_ACT\_PLANE$  
Bit 11: $AC\_MEAS\_FRAME\_SELECT$  
Bit 12: $AC\_MEAS\_TYPE$  
Bit 13: $AC\_MEAS\_FINE\_TRANS$  
Bit 14: $AA\_MEAS\_SETANGLE[\text{axis}]$  
Bit 15: $AC\_MEAS\_SCALEUNIT$  
Bit 16: $AC\_MEAS\_TOOL\_MASK$  
Bit 17: $AC\_MEAS\_P1\_COORD$  
Bit 18: $AC\_MEAS\_P2\_COORD$  
Bit 19: $AC\_MEAS\_P3\_COORD$  
Bit 20: $AC\_MEAS\_P4\_COORD$  
Bit 21: $AC\_MEAS\_SET\_COORD$  
Bit 22: $AC\_MEAS\_CHSFR$  
Bit 23: $AC\_MEAS\_NCBFR$  
Bit 24: $AC\_MEAS\_CHBFR$  
Bit 25: $AC\_MEAS\_UIFR$  
Bit 26: $AC\_MEAS\_PF\_FRAME$ | R   | W   | 6 |
| $AC\_MEAS\_FRAME$ | FRAME | $AC\_MEAS\_FRAME$  
Result frame for part gauging. | R | W | 6 |
| $AC\_MEAS\_WP\_ANGLE$ | REAL | $AC\_MEAS\_WP\_ANGLE$  
Calculated part position angle for part gauging. | R | | 6 |
| $AC\_MEAS\_CORNER\_ANGLE$ | REAL | $AC\_MEAS\_CORNER\_ANGLE$  
Calculated cutting angle of the corner for part gauging. | R | | 6 |
| $AC\_MEAS\_DIAMETER$ | REAL | $AC\_MEAS\_DIAMETER$  
Calculated diameter for part and tool gauging. | R | | 6 |
| $AC\_MEAS\_TOOL\_LENGTH$ | REAL | $AC\_MEAS\_TOOL\_LENGTH$  
Calculated tool length for tool gauging. | R | | 6 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{AC_MEAS_RESULTS}$</td>
<td>REAL</td>
<td>$R_0 = \text{AC_MEAS_RESULTS}[0]$ Measured results</td>
<td>R</td>
<td>6</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>$\text{AC_MEAS_SCALEUNIT}$</td>
<td>INT</td>
<td>Unit of measurement in accordance with the configuration $\text{AC_MEAS_SCALEUNIT} = 0$ Unit of measurement relative to the active G code G70/G700/G71/G710</td>
<td>R, W</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$\text{P_CHANNO}$</td>
<td>INT</td>
<td>Query the current channel number.</td>
<td>R</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$\text{AC_SERUPRO}$</td>
<td>INT</td>
<td>$\text{AC_SERUPRO}$ Query whether search run type SERUPRO is active. (SERUPRO: &quot;Search run via program testing&quot;) Possible to use in SYNACTs and in the parts program $\text{P_CHANNO} = 0$ SERUPRO search run type not active $\text{P_CHANNO} = 1$ SERUPRO search run type is active</td>
<td>R</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

#### 15.2.44 Positions

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{P_EP}$</td>
<td>REAL</td>
<td>$\text{P_EP}[X]$ The system variable $\text{P_EP}$ always returns the current WCS setpoint position in the interpreter. The numerical value is not inevitably the same as the programmed value in the parts program. In the following situations there are differences - during incremental programming - when changing the WCS by frame or tool selection If an ASUB starts after a block search with calculation, this event will synchronize positions in the interpreter. $\text{P_EP}$ then returns in ASUB the position at which the axes actually are. The search run position that is picked up can be queried via system variable $\text{AC_RETPPOINT}$. Axes: channel axis</td>
<td>R</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_EPM</td>
<td>REAL</td>
<td>$P_EPM[X]$ Current MCS position in the interpreter (also see $P_EP$). Axes: channel axis</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_APR</td>
<td>REAL</td>
<td>$P_APR[X]$ Position of axis in the workpiece coordinate system at the start of the approach motion for soft approach to the contour. Axes: channel axis</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_AEP</td>
<td>REAL</td>
<td>$P_AEP[X]$ Approach point: first contour point in the workpiece coordinate system for soft approach to contour. Axes: channel axis</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_POLF</td>
<td>REAL</td>
<td>$P_POLF[X]$ X: Axis returns the programmed return position of the axis. Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_POLF_VALID</td>
<td>INT</td>
<td>$P_POLF_VALID[X]$ X: Axis 0: no axis return programmed 1: return programmed in abs. position 2: return programmed as distance Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_IW</td>
<td>REAL</td>
<td>$AA_IW[X]$ Actual value in workpiece coordinate system (WCS). Axes: channel axis</td>
<td>RS R 2</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AA_REPOS_DELAY</td>
<td>BOOL</td>
<td>$AA_REPOS_DELAY[X]$ TRUE: For this axis REPOS suppression is just active. FALSE: otherwise Axes: channel axis</td>
<td>RS R 6+</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$AA_IEN</td>
<td>REAL</td>
<td>$AA_IEN[X]$ Actual value in the settable origin system (SOS). Axes: channel axis</td>
<td>RS R 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_IBN</td>
<td>REAL</td>
<td>$AA_IBN[X]$ Actual value in the basic origin system (BOS). Axes: channel axis</td>
<td>RS R 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_IB</td>
<td>REAL</td>
<td>$AA_IB[X]$ Actual value in basic coordinate system (BCS). Axes: channel axis</td>
<td>RS R 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_IM</td>
<td>REAL</td>
<td>$AA_IM[X]$ Actual value in machine coordinate system (MCS). Axes: geometry axis, channel axis, machine axis</td>
<td>RS R 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.45 Indexing axes

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$AA_{ACT_INDE_X_AX_POS_NO}$</td>
<td>INT</td>
<td>$$AA_{INDEX_AX_POS_NO}[X]$ 0: No indexing axis, therefore no indexing position available. &gt; 0: Number of indexing position last reached or crossed Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>$$AA_{PROG_INDE_X_AX_POS_NO}$</td>
<td>INT</td>
<td>$$AA_{PROG_INDEX_AX_POS_NO}[X]$ 0: No indexing axis, thus no indexing position available or indexing axis currently not approaching an indexing position &gt; 0: Number of programmed indexing position Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

### 15.2.46 Encoder values

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$AA_{ENC_ACTIVE}$</td>
<td>BOOL</td>
<td>$$AA_{ENC_ACTIVE}[X]$ Active measuring system is operating below encoder limit frequency Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$$AA_{ENC1_ACTIVE}$</td>
<td>BOOL</td>
<td>$$AA_{ENC1_ACTIVE}[X]$ Encoder 1 is operating below encoder limit frequency Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$$AA_{ENC2_ACTIVE}$</td>
<td>BOOL</td>
<td>$$AA_{ENC2_ACTIVE}[X]$ Encoder 2 is operating below encoder limit frequency Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$$VA_{IM}$</td>
<td>REAL</td>
<td>$$VA_{IM}[X]$ Encoder actual value in machine coordinate system (measured on active measuring system), actual value compensations are corrected (leadscrew error compensation, backlash compensation, quadrant error compensation) With active spindle/axis disable, the default return is the current setpoint. If the actual value is then returned, BIT3 must be set in $$MA_{MISC_FUNCTION_MASK}$.</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$VA_{IM1}$</td>
<td>REAL</td>
<td>$VA_{IM1}[X]$ Actual value in the machine coordinate system (measured Encoder 1), compensations are corrected With active spindle/axis disable the default return is the current setpoint. If the actual value is then returned, BIT3 must be set in $MA_{MISC_FUNCTION_MASK}$.</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$VA_{IM2}$</td>
<td>REAL</td>
<td>$VA_{IM2}[X]$ Actual value in the machine coordinate system (measured Encoder 2), compensations are corrected With active spindle/axis disable the default return is the current setpoint. If the actual value is then returned, BIT3 must be set in $MA_{MISC_FUNCTION_MASK}$.</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AA_{MW}$</td>
<td>REAL</td>
<td>$AA_{MW}[X]$ Measured value in workpiece coordinate system Axes: channel axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MM}$</td>
<td>REAL</td>
<td>$AA_{MM}[X]$ Measured value in machine coordinate system</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MW1}$</td>
<td>REAL</td>
<td>$AA_{MW1}[X]$ Measurement result of axial measurement Trigger event 1 in WCS Axes: channel axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MW2}$</td>
<td>REAL</td>
<td>$AA_{MW2}[X]$ Measurement result of axial measurement Trigger event 2 in WCS Axes: channel axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MW3}$</td>
<td>REAL</td>
<td>$AA_{MW3}[X]$ Measurement result of axial measurement Trigger event 3 in WCS Axes: channel axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MW4}$</td>
<td>REAL</td>
<td>$AA_{MW4}[X]$ Measurement result of axial measurement Trigger event 4 in WCS Axes: channel axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>

### 15.2.47 Axial measurement

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_{MM1}$</td>
<td>REAL</td>
<td>$AA_{MM1}[X]$ Measurement result of axial measurement Trigger event 1 in MCS</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MM2}$</td>
<td>REAL</td>
<td>$AA_{MM2}[X]$ Measurement result of axial measurement Trigger event 2 in MCS</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA_{MM3}$</td>
<td>REAL</td>
<td>$AA_{MM3}[X]$ Measurement result of axial measurement Trigger event 3 in MCS</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA.MM4</td>
<td>REAL</td>
<td>$AA.MM4[X] Measurement result of axial measurement Trigger event 4 in MCS</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA.MEAACT</td>
<td>BOOL</td>
<td>$AA.MEAACT[X] Value is TRUE if axial measurement is active for X Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 15.2.48 Offsets

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AC.DRF</td>
<td>REAL</td>
<td>$AC.DRF[X] DRF offset Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC.PRESET</td>
<td>REAL</td>
<td>$AC.PRESET[X] Last given PRESET value Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA.ETRANS</td>
<td>REAL</td>
<td>$AA.ETRANS[X] External zero offset Axes: channel axis</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA.MEAS.P1._VALID</td>
<td>INT</td>
<td>$AA.MEAS_P1._VALID[X] = 1 Writing the actual axis value to the 1st measuring point Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA.MEAS.P2._VALID</td>
<td>INT</td>
<td>$AA.MEAS_P2._VALID[X] = 1 Writing the actual axis value to the 2nd measuring point Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA.MEAS.P3._VALID</td>
<td>INT</td>
<td>$AA.MEAS_P3._VALID[X] = 1 Writing the actual axis value to the 3rd measuring point Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA.MEAS.P4._VALID</td>
<td>INT</td>
<td>$AA.MEAS_P4._VALID[X] = 1 Writing the actual axis value to the 4th measuring point Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$AA.MEAS.POINT1</td>
<td>REAL</td>
<td>$AA.MEAS.POINT1[x] = $AA.IW[x] $AA.MEAS.POINT1[y] = $AA.IW[y] $AA.MEAS.POINT1[z] = $AA.IW[z] First measuring point for part and tool gauging Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA.MEAS.POINT2</td>
<td>REAL</td>
<td>$AA.MEAS.POINT2[x] = $AA.IW[x] $AA.MEAS.POINT2[y] = $AA.IW[y] $AA.MEAS.POINT2[z] = $AA.IW[z] Second measuring point for part and tool gauging Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### List of System Variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $\text{AA\_MEAS\_POINT3}$ | REAL   | $\text{AA\_MEAS\_POINT3}\[x\] = \$AA\_IW\[x\]$  
$\text{AA\_MEAS\_POINT3}\[y\] = \$AA\_IW\[y\]$  
$\text{AA\_MEAS\_POINT3}\[z\] = \$AA\_IW\[z\]$  
Third measuring point for part and tool gauging.  
Axes: geometry axis, channel axis, machine axis | R   | W | 6 | 1 |
| $\text{AA\_MEAS\_POINT4}$ | REAL   | $\text{AA\_MEAS\_POINT4}\[x\] = \$AA\_IW\[x\]$  
$\text{AA\_MEAS\_POINT4}\[y\] = \$AA\_IW\[y\]$  
$\text{AA\_MEAS\_POINT4}\[z\] = \$AA\_IW\[z\]$  
Fourth measuring point for part and tool gauging.  
Axes: geometry axis, channel axis, machine axis | R   | W | 6 | 1 |
| $\text{AA\_MEAS\_SP\_VALID}$ | INT    | $\text{AA\_MEAS\_SP\_VALID}\[X\] = 0$  
Invalidating the x-axis setpoint for part and tool gauging.  
Axes: geometry axis, channel axis, machine axis | R   | W | 6 | 1 |
| $\text{AA\_MEAS\_SETPOINT}$ | REAL   | $\text{AA\_MEAS\_SETPOINT}\[x\] = 0.0$  
$\text{AA\_MEAS\_SETPOINT}\[y\] = 0.0$  
$\text{AA\_MEAS\_SETPOINT}\[z\] = 0.0$  
Setpoint position for part and tool gauging.  
Axes: geometry axis, channel axis, machine axis | R   | W | 6 | 1 |
| $\text{AA\_MEAS\_SETANGLE}$ | REAL   | $\text{AA\_MEAS\_SETANGLE}\[x\] = 0.0$  
$\text{AA\_MEAS\_SETANGLE}\[y\] = 0.0$  
$\text{AA\_MEAS\_SETANGLE}\[z\] = 0.0$  
Setpoint angle for part and tool gauging.  
Axes: geometry axis, channel axis, machine axis | R   | W | 6 | 4 |
| $\text{AA\_OFF}$ | REAL   | $\$AA\_OFF\[X\]$  
Overlaid motion for programmed axis  
Axes: geometry axis, channel axis, machine axis | RS  | W | R | W | 3 |
| $\text{AA\_OFF\_LIMIT}$ | INT    | $\$AA\_OFF\_LIMIT\[axis\]$  
Limit value for axial offset $\$AA\_OFF\[axis\]$  
0: Limit value not reached  
1: Limit value reached in positive axis direction  
-1: Limit value reached in negative axis direction  
Axes: geometry axis, channel axis, machine axis | RS  | R | 4 |
| $\text{AA\_OFF\_VAL}$ | REAL   | $\$AA\_OFF\_VAL\[axis\]$  
Integrated value of the overlaid movement for one axis.  
An overlaid movement can be undone by using the negative value of these variables.  
For example $\$AA\_OFF\[axis\] = -$\$AA\_OFF\_VAL\[axis\]$  
Axes: geometry axis, channel axis, machine axis | RS  | R | 6 | 1 |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\textit{AC}_\text{RETPINT}$</td>
<td>REAL</td>
<td>$\textit{AC}_\text{RETPINT}[X]$&lt;br&gt;\textit{AC}_\text{RETPINT[]} returns the WCS position of an axis at which an ASUB was started. Then repositioning to this position can take place in the ASUB.&lt;br&gt;\textit{AC}_\text{RETPINT[]} is set by calculation. \textit{AC}_\text{RETPINT} returns the search run position that has been picked up.&lt;br&gt;Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\textit{AA}_\text{TOFF}$</td>
<td>REAL</td>
<td>$\textit{AA}_\text{TOFF}$[geometry axis]&lt;br&gt;Overlaid value in the tool coordinate system.&lt;br&gt;Axes: geometry axis</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$\textit{AA}_\text{TOFF_VAL}$</td>
<td>REAL</td>
<td>$\textit{AA}_\text{TOFF_VAL}$[geometry axis]&lt;br&gt;Overlaid value in the tool coordinate system (integrated).&lt;br&gt;Axes: geometry axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\textit{AA}_\text{TOFF_LIMIT}$</td>
<td>INT</td>
<td>$\textit{AA}_\text{TOFF_LIMIT}$[geo axis]&lt;br&gt;Limit value for axial offset $\textit{AA}_\text{TOFF}$[geo axis]&lt;br&gt;0: Limit value not reached&lt;br&gt;1: Limit value reached in positive axis direction&lt;br&gt;-1: Limit value reached in negative axis direction&lt;br&gt;Axes: geometry axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\textit{AA}_\text{TOFF_PREP_DIFF}$</td>
<td>REAL</td>
<td>$\textit{AA}_\text{TOFF_PREP_DIFF}$[geometry axis]&lt;br&gt;Difference value of the override in the tool coordinate system between the main run and preprocessing.&lt;br&gt;Axes: geometry axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$\textit{AA}_\text{SOFTENDP}$</td>
<td>REAL</td>
<td>$\textit{AA}_\text{SOFTENDP}[X]$&lt;br&gt;Software end position, positive direction&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$\textit{AA}_\text{SOFTENDN}$</td>
<td>REAL</td>
<td>$\textit{AA}_\text{SOFTENDN}[X]$&lt;br&gt;Software end position, negative direction&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.49 Axial paths

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_DTBW$</td>
<td>REAL</td>
<td>$AA_DTBW[X]$ axial path from start of block in the workpiece coordinate system for positioning and synchronized axes for motion synchronized action. The programmed position is decisive for computing the path; if the axis is a coupling axis, the position part that results from axis coupling is not considered here. This variable can only be accessed from synchronized actions. Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AA_DTBB$</td>
<td>REAL</td>
<td>$AA_DTBB[X]$ Axial distance from start of block in basic coordinate system for positioning and synchronized axes with motion-synchronized actions. The programmed position is decisive for computing the path; if the axis is a coupling axis, the position part that results from axis coupling is not considered here. This variable can only be accessed from synchronized actions. Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AA_DTEW$</td>
<td>REAL</td>
<td>$AA_DTEW[X]$ Axial distance to end of block in workpiece coordinate system for positioning and synchronized axes with motion-synchronized actions. The programmed position is decisive for computing the path; if the axis is a coupling axis, the position part that results from axis coupling is not considered here. This variable can only be accessed from synchronized actions. Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AA_DTEB$</td>
<td>REAL</td>
<td>$AA_DTEB[X]$ Axial distance to end of block in basic coordinate system for positioning and synchronized axes with motion-synchronized actions. The programmed position is decisive for computing the path; if the axis is a coupling axis, the position part that results from axis coupling is not considered here. This variable can only be accessed from synchronized actions. Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2.50 Oscillation

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$AA_DTEPW</td>
<td>REAL</td>
<td>$$AA_DTEPW[X] Axial distance-to-go for infeed oscillation in workpiece coordinate system This variable can only be accessed from synchronized actions Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$$AA_DTEPB</td>
<td>REAL</td>
<td>$$AA_DTEPB[X] Axial distance-to-go for infeed oscillation in basic coordinate system This variable can only be accessed from synchronized actions Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$$AA_OSCILL_REVERSE_POS1</td>
<td>REAL</td>
<td>$$AA_OSCILL_REVERSE_POS1[X] Current reversal position 1 for oscillation In synchronized actions, the setting data value $$SA_OSCILL_REVERSE_POS1 is evaluated online This variable can only be accessed from synchronized actions Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$$AA_OSCILL_REVERSE_POS2</td>
<td>REAL</td>
<td>$$AA_OSCILL_REVERSE_POS2[X] Current reversal position 2 for oscillation In synchronized actions, the setting data value $$SA_OSCILL_REVERSE_POS2 is evaluated online This variable can only be accessed from synchronized actions Axes: channel axis</td>
<td>RS</td>
<td>R</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$$AA_DELT</td>
<td>REAL</td>
<td>$$AA_DELT[X] Stored axial distance-to-go path in the workpiece coordinate system subsequent to deletion of the residual distance during synchronized motion actions Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### 15.2.51 Axial velocities

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_FA</td>
<td>REAL</td>
<td>$P_FA[X]$ Last programmed axial feedrate Axes: channel axis</td>
<td></td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
### List of System Variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AA_OVR   | REAL   | Axial override for motion-synchronized actions
Multiplicative override component acting in addition to the
user OV, programmed OV and transformation OV.
The value is limited to max. 200%. If a value < 0.0 is entered, 0 is assumed and alarm 14756 reported.
Must be rewritten in every interpolator cycle, otherwise the value is 100%.
The spindle override is changed with $AA_OVR[S1].
This variable can only be accessed from motion-synchronized actions
Axes: channel axis                                                                                         | R | W | 2 |
| $AA_VC    | REAL   | Additive axial feed compensation for motion-synchronized actions
It must be rewritten in every interpolator cycle, otherwise the value is 0.
With an override of 0, the compensation value has no effect, otherwise the override has no impact on the compensation value.
The compensation value cannot make the total feedrate negative.
The upper value is limited such that the maximum axis velocities and accelerations are not exceeded.
The computation of the other feedrate components is not affected by $AA_VC.
The override defined by machine data
$MN_OVR_FACTOR_LIMIT_BIN, $MN_OVR_FACTOR_FEEDRATE[30],
$MN_OVR_FACTOR_AX_SPEED[30] and
$MN_OVR_FACTOR_SPIND_SPEED
Override values cannot be exceeded. The additive feedrate override is limited such that the resulting feedrate does not exceed the maximum override value of the programmed feedrate.
This variable can only be accessed from synchronized actions
Axes: channel axis                                                                                         | R | W | 2 |
| $AA_VACTB | REAL   | Axis velocity in the base coordinate system
This variable can only be accessed from synchronized actions
Axes: channel axis                                                                                         | RS | R | 2 |
| $AA_VACTW | REAL   | Axis velocity in workpiece coordinate system
This variable can only be accessed from synchronized actions
Axes: channel axis                                                                                         | RS | R | 2 |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync O S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_VACTM</td>
<td>REAL</td>
<td>$AA_VACTM{[X]}&lt;br&gt;Axis velocity, setpoint-related in machine coordinate system&lt;br&gt;Can also be read for replacement and PLC axes&lt;br&gt;This variable can only be accessed from synchronized actions&lt;br&gt;Axes: channel axis</td>
<td>RS</td>
<td>R 4</td>
</tr>
<tr>
<td>$VA_VACTM</td>
<td>REAL</td>
<td>$VA_VACTM{[X]}&lt;br&gt;Axis velocity, actual value-related in machine coordinate system&lt;br&gt;The variable returns an undefined value if the encoder limit frequency is exceeded&lt;br&gt;This variable can only be accessed from synchronized actions&lt;br&gt;Axes: channel axis</td>
<td>RS</td>
<td>R 4</td>
</tr>
</tbody>
</table>

**15.2.52 Drive data**

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync O S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_LOAD</td>
<td>REAL</td>
<td>$AA_LOAD{[X]}&lt;br&gt;Drive capacity utilization as % (for 611D or PROFIBUS only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 2</td>
</tr>
<tr>
<td>$VA_LOAD</td>
<td>REAL</td>
<td>$VA_LOAD{[X]}&lt;br&gt;Drive capacity utilization as % (for 611D or PROFIBUS only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 5</td>
</tr>
<tr>
<td>$AA_TORQUE</td>
<td>REAL</td>
<td>$AA_TORQUE{[X]}&lt;br&gt;Drive torque setpoint in Nm (for 611D only)&lt;br&gt;Force actual value in N (611D-HLA only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 2</td>
</tr>
<tr>
<td>$VA_TORQUE</td>
<td>REAL</td>
<td>$VA_TORQUE{[X]}&lt;br&gt;Drive torque setpoint in Nm (for 611D only)&lt;br&gt;Force actual value in N (611D-HLA only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 5</td>
</tr>
<tr>
<td>$AA_POWER</td>
<td>REAL</td>
<td>$AA_POWER{[x]}&lt;br&gt;Drive active power in W (for 611D only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 2</td>
</tr>
<tr>
<td>$VA_POWER</td>
<td>REAL</td>
<td>$VA_POWER{[x]}&lt;br&gt;Drive active power in W (for 611D only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 5</td>
</tr>
<tr>
<td>$AA_CURR</td>
<td>REAL</td>
<td>$AA_CURR{[X]}&lt;br&gt;Actual current value of axis or spindle in A (for 611D only)&lt;br&gt;Axes: channel axis, machine axis</td>
<td>RS</td>
<td>R 2</td>
</tr>
</tbody>
</table>
### 15.2.52 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $VA_CURR       | REAL   | $VA_CURR[X]  
Actual current value of axis or spindle in A (for 611D only)  
Axes: channel axis, machine axis | RS R      | 5   |   |   |
| $VA_DIST_TORQUE| REAL   | $VA_DIST_TORQUE[X]  
Disturbing torque/max. motor torque (output of disturbance torque observer)  
Axes: channel axis, machine axis | RS R      | 6   | 3 |   |
| $VA_VALVELIFT  | REAL   | $VA_VALVELIFT[X]  
Actual valve stroke in mm (for 611D hydraulics only)  
Axes: channel axis, machine axis | RS R      | 5   | 1 |   |
| $VA_PRESSURE_A | REAL   | $VA_PRESSURE_A[X]  
Pressure on A side of cylinder in bar (for 611D hydraulics only)  
Axes: channel axis, machine axis | RS R      | 5   | 1 |   |
| $VA_PRESSURE_B | REAL   | $VA_PRESSURE_B[X]  
Pressure on B side of cylinder in bar (for 611D hydraulics only)  
Axes: channel axis, machine axis | RS R      | 5   | 1 |   |
| $VA_DP_ACT_TEL | INT    | $VA_DP_ACT_TEL[b,a]  
\text{b: Word index (16-bit access) in PROFIBUS frame}  
\text{a: Machine axis}  
\text{b: Word index in PROFIBUS actual value frame}  
Axes: geometry axis, channel axis, machine axis | RS R      | 6   | 4 |   |

### 15.2.53 Axis statuses

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $AA_STAT     | INT  | $AA_STAT[X]  
Axis status:  
0: No axis status available  
1: Traversing motion in progress  
2: Axis has reached IPO end  
3: Axis in position (exact stop coarse) for all axes  
4: Axis in position (exact stop fine) for all axes  
Axes: geometry axis, channel axis, machine axis | RS R      | 4   |   |   |
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_SNGLAX_STAT</td>
<td>INT</td>
<td>$AA_SNGLAX_STAT[X]&lt;br&gt;Axis status:&lt;br&gt;0: Axis is not a single axis&lt;br&gt;1: Single axis in reset&lt;br&gt;2: Single axis is ended&lt;br&gt;3: Single axis is interrupted&lt;br&gt;4: Single axis is active&lt;br&gt;5: Single axis alarm pending&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$AA_REF</td>
<td>INT</td>
<td>$AA_REF[X]&lt;br&gt;Axis status:&lt;br&gt;0: Axis is not referenced&lt;br&gt;1: Axis is referenced&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td>.</td>
</tr>
<tr>
<td>$AA_TYP</td>
<td>INT</td>
<td>$AA_TYP[X]&lt;br&gt;Axis type:&lt;br&gt;0: Axis on other channel&lt;br&gt;1: Channel axis of local channel&lt;br&gt;2: Neutral axis&lt;br&gt;3: PLC axis&lt;br&gt;4: Oscillating axis&lt;br&gt;5: Neutral axis currently traversing in JOG mode&lt;br&gt;6: Master value linked following axis&lt;br&gt;7: Coupled motion following axis&lt;br&gt;8: Command axis&lt;br&gt;9: Compile cycle axis&lt;br&gt;10: Linked slave axis (master/slave function)&lt;br&gt;Axes: geometry axis, channel axis</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td>.</td>
</tr>
</tbody>
</table>

#### 15.2.54 Master/slave links

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_MASL_STAT</td>
<td>INT</td>
<td>The current status of a master/slave link.&lt;br&gt;Value 0: Axis is no slave axis, or no active link.&lt;br&gt;Value &gt; 0: Active link; the associated machine axis number of the master axis is returned.&lt;br&gt;$AA_MASL_STAT[X]&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$P_SEARCH_MASLC</td>
<td>INT</td>
<td>$P_SEARCH_MASLC[&lt;axis identifier&gt;]&lt;br&gt;The current status of a master/slave link was changed in search mode.&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

#### 15.2.55 Travel to fixed stop

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_SEARCH_MASLD</td>
<td>REAL</td>
<td>In search mode, the positional offset determined when closing the link between the master and the slave axis. Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$AA_FXS</td>
<td>INT</td>
<td>Setpoint status state &quot;travel to fixed stop&quot; 0: Axis not at fixed stop 1: Fixed stop successfully approached 2: Fixed stop approach has failed 3: Travel to fixed stop selection active 4: Fixed stop detected 5: Travel to fixed stop deselection active Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>W</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$VA_FXS</td>
<td>INT</td>
<td>Actual status state &quot;travel to fixed stop&quot; 0: Axis not at fixed stop 1: Fixed stop successfully approached 2: Fixed stop approach has failed 3: Travel to fixed stop selection active 4: Fixed stop detected 5: Travel to fixed stop deselection active Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$VA_FXS_INFO</td>
<td>INT</td>
<td>Additional information for &quot;travel to fixed stop&quot; when $VA_FXS_INFO[X]=2 0: No additional information available 1: No approach motion programmed 2: Programmed end position reached, motion ended 3: Abort caused by NC reset (pushbutton reset) 4: Exit fixed stop window 5: Drive has refused torque reduction 6: PLC has canceled enabling Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$VA_TORQUE_AT_LIMIT</td>
<td>INT</td>
<td>Status &quot;Torque limit reached&quot; 0: Torque limit not yet reached 1: Torque limit reached In the digital 611D systems, the drive returns the status indicating whether the programmed torque limit has been reached. Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_FOC</td>
<td>INT</td>
<td>$AA_FOC[X]$ Setpoint status state &quot;ForceControl&quot;</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: ForceControl not active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: ForceControl modal active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Block-related ForceControl active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$VA_FOC</td>
<td>INT</td>
<td>$VA_FOC[X]$ Actual status state &quot;ForceControl&quot;</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: ForceControl not active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: ForceControl modal active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Block-related ForceControl active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_COUP_ACT</td>
<td>INT</td>
<td>$AA_COUP_ACT[SPI(2)] Current coupling status of following spindle/following axis:</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Axis/spindle is not coupled to a leading spindle/leading axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: Tangential follow-up of axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4: synchronized spindle coupling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8: Axis is trailing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>16: Following axis of master value coupling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The respective values apply to one coupling. If several couplings are active for a following axis, this is represented by the sum of the relevant numerical values. Axes: geometry axis, channel axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 15.2.56 Electronic gear

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA_EG_SYNFA</td>
<td>REAL</td>
<td>$AA_EG_SYNFA[a]$ a: Following axis Synchronized position of the following axis Axes: geometry axis, channel axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_EG_BC</td>
<td>STRING</td>
<td>$P_EG_BC[a]$ Block change condition for EGONSYN, EGON, WAITC. 2nd dimension for TYPE_STRING is automatically MAXSTRINGLEN</td>
<td>R</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$AA_EG_NUM_</td>
<td>INT</td>
<td>$AA_EG_NUM_LA[a]$ a: Following axis Number of leading axes specified with EGDEF Axes: geometry axis, channel axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$VA_EG_SYNC</td>
<td>REAL</td>
<td>$VA_EG_SYNCDIFF[a]$ a: Following axis Synchronized run difference Axes: geometry axis, channel axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Tables

#### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$VA_EG_SYNC$</td>
<td>REAL</td>
<td>$VA_EG_SYNCDIFF_S[a]$</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Following axis</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Synchronized run difference with sign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n: Index for leading axis</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Following axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identifier for nth leading axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 15.2.57 Leading value coupling

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_LEAD_SP$</td>
<td>REAL</td>
<td>$AA_LEAD_SP[LW]$</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simulated master value - position</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$AA_LEAD_SV$</td>
<td>REAL</td>
<td>$AA_LEAD_SV[LW]$</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simulated master value - velocity</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$AA_LEAD_P_TURN$</td>
<td>REAL</td>
<td>$AA_LEAD_P_TURN[LW]$</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Current leading value positions lost through modulo reduction.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The actual master value position (which the control uses for internal calculation) is $AA_LEAD_P[LW] + AA_LEAD_P_TURN[LW]$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If MV is a modulo axis, $AA_LEAD_P_TURN$ is an integral multiple of $MA_MODULO_RANGE$.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If MV is not a modulo axis, $AA_LEAD_P_TURN$ is always 0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 1: $MA_MODULO_RANGE[LW]=360$ $AA_LEAD_P[LW]=290$ $AA_LEAD_P_TURN[LW]=720$ The actual master value position (used internally by the control in calculations) is 1010.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_LEAD_P$</td>
<td>REAL</td>
<td>$AA_LEAD_P[LW]$</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Current master value - position (modulo-reduced)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If MV is a modulo axis, the following always applies: 0 &lt;= $AA_LEAD_P[LW] &lt;= $MA_MODULO_RANGE[LW]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_LEAD_V</td>
<td>REAL</td>
<td>Current master value - velocity</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AA_SYNC</td>
<td>INT</td>
<td>Coupling status of following axis in master value coupling&lt;br&gt;0 =&gt; No synchronism&lt;br&gt;1 =&gt; Coarse synchronism&lt;br&gt;2 =&gt; Fine synchronism&lt;br&gt;3 =&gt; Synchronized run coarse and fine&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$AA_IN_SYNC</td>
<td>INT</td>
<td>Synchronization status of the following axis for master value coupling and ELG&lt;br&gt;1 =&gt; Synchronization in progress, i.e. following axis is synchronized out&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>$P_COUP_OFFS</td>
<td>REAL</td>
<td>Programmed positional offset for the synchronous spindle (following spindle)</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_COUP_OF_FS</td>
<td>REAL</td>
<td>Positional offset for synchronous spindle (following spindle) setpoint value viewpoint&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$VA_COUP_OF_FS</td>
<td>REAL</td>
<td>Positional offset for synchronous spindle (following spindle) actual value viewpoint</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AA_SCTRACE</td>
<td>BOOL</td>
<td>Write: Initiate IPO trigger for servo trace&lt;br&gt;0: No action&lt;br&gt;10: Initiate trigger&lt;br&gt;Read: Always 0, as the trigger cannot be read back&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$VA_DPE</td>
<td>BOOL</td>
<td>Status of power enable of a machine axis&lt;br&gt;Axes: Machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>$AA_ACC</td>
<td>REAL</td>
<td>Current acceleration value of axis with 1-axis interpolation. $AA_ACC = $MA_MAX_AX_ACCEL * progr. acceleration correction</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

---

### 15.2.58 Synchronized spindle

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_COUP_OFFS</td>
<td>REAL</td>
<td>Programmed positional offset for the synchronous spindle (following spindle)</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_COUP_OF_FS</td>
<td>REAL</td>
<td>Positional offset for synchronous spindle (following spindle) setpoint value viewpoint&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$VA_COUP_OF_FS</td>
<td>REAL</td>
<td>Positional offset for synchronous spindle (following spindle) actual value viewpoint</td>
<td>RS</td>
<td>R</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$AA_SCTRACE</td>
<td>BOOL</td>
<td>Write: Initiate IPO trigger for servo trace&lt;br&gt;0: No action&lt;br&gt;10: Initiate trigger&lt;br&gt;Read: Always 0, as the trigger cannot be read back&lt;br&gt;Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>$VA_DPE</td>
<td>BOOL</td>
<td>Status of power enable of a machine axis&lt;br&gt;Axes: Machine axis</td>
<td>RS</td>
<td>R</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>$AA_ACC</td>
<td>REAL</td>
<td>Current acceleration value of axis with 1-axis interpolation. $AA_ACC = $MA_MAX_AX_ACCEL * progr. acceleration correction</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PA_{ACCLIMA}$</td>
<td>INT</td>
<td>$PA_{ACCLIMA}$ Acceleration override set in preprocessing with ACCLIMA Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>$PA_{VELOLIMA}$</td>
<td>INT</td>
<td>$PA_{VELOLIMA}$ Velocity override set in preprocessing with VELOLIMA Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>$PA_{JERKLIMA}$</td>
<td>INT</td>
<td>$PA_{JERKLIMA}$ Jerk override set in preprocessing with JERKLIMA Axes: geometry axis, channel axis, machine axis</td>
<td>R</td>
<td>6</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>$AA_{ACCLIMA}$</td>
<td>INT</td>
<td>$AA_{ACCLIMA}$ Acceleration override set in main run with ACCLIMA Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$AA_{VELOLIMA}$</td>
<td>INT</td>
<td>$AA_{VELOLIMA}$ Velocity override set in main run with VELOLIMA Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$AA_{JERKLIMA}$</td>
<td>INT</td>
<td>$AA_{JERKLIMA}$ Jerk override set in main run with JERKLIMA Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td>.</td>
</tr>
<tr>
<td>$AA_{MOTEND}$</td>
<td>INT</td>
<td>$AA_{MOTEND}$ Current motion end criterion at 1-axis interpolation 1 = Motion end at exact stop FINE 2 = Motion end at exact stop COARSE 3 = Motion end at exact stop, IPO stop 4 = Block change in braking ramp of axis motion 5 = Block change in braking ramp of axis motion with tolerance window with regard to setpoint 6 = Block change in braking ramp of axis motion with tolerance window with regard to actual value Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$AA_{SCPAR}$</td>
<td>INT</td>
<td>$AA_{SCPAR}$ Read current servo parameter set Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$AA_{ESR_STAT}$</td>
<td>INT</td>
<td>$AA_{ESR_STAT}[X]$ Status of &quot;Extended stop and retract&quot;, bit-coded: BIT0: Generator operation triggered BIT1: Retraction triggered BIT2: Ext. stop triggered BIT3: DC link undervoltage BIT4: Generator minimum speed Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>R</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$AA_{ESR_ENABLE}$</td>
<td>BOOL</td>
<td>$AA_{ESR_ENABLE}[X] = 1$ Enable &quot;Extended stop and retract&quot; Axes: geometry axis, channel axis, machine axis</td>
<td>RS</td>
<td>WS</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
### 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AA_ESR_TRIGGER</td>
<td>BOOL</td>
<td>$AA_ESR_TRIGGER = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Trigger “NC-driven ESR” for PLC controlled axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: channel axis</td>
<td>RW 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_POLFA</td>
<td>REAL</td>
<td>$AA_POLFA[X]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X: Single axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>returns the programmed return position of the single axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis</td>
<td>RS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_POLFA_VALID</td>
<td>INT</td>
<td>$AA_POLFA_VALID[X]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X: Retraction programmed for this single axis, returns</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Single axis retraction not programmed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Retraction programmed as position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Retraction programmed as distance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_ALARM_STAT</td>
<td>INT</td>
<td>(Selected) alarm reactions for synchronized actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(SYNFCT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: channel axis</td>
<td>RS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AN_AXCTSWA</td>
<td>BOOL</td>
<td>EVERY $AN_AXCTSWA[n] == TRUE DO M99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: an axis container rotation is currently being executed on the container with the axis container name n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: No active axis container rotation is active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AN_AXCTAS</td>
<td>INT</td>
<td>Read:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axis container rotation current rotation:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of slots the axis container has currently been advanced is indicated for the axis container with axis container name n.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value range is from 0 to the maximum number of assigned slots in axis container -1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_AXCTSWA</td>
<td>BOOL</td>
<td>IF $AC_AXCTSWA[n] == TRUE GOTOB MARK1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: The channel has enabled axis container rotation for the axis container name n and the rotation has not yet been completed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: The axis container rotation is terminated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_EG_TYPE</td>
<td>INT</td>
<td>$AA_EG_TYPE[a,b]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Following axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: Leading axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type of coupling for leading axis b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Actual-value coupling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Setpoint linkage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA_EG_NUMERA</td>
<td>REAL</td>
<td>$AA_EG_NUMERA[a,b]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: Following axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: Leading axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numerator of coupling factor for leading axis b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Axes: geometry axis, channel axis, machine axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 15.2 List of system variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync O S</th>
</tr>
</thead>
</table>
| $\text{AA}_\text{EG\_DENOM}$ | REAL     | $\text{AA}_\text{EG\_DENOM}[a,b]$  
  a: Following axis  
  b: Leading axis  
  Denominator of coupling factor for leading axis b  
  Axes: geometry axis, channel axis, machine axis | RS         | R         | O 6       | S 1       |
| $\text{AA}_\text{EG\_SYN}$   | REAL     | $\text{AA}_\text{EG\_SYN}[a,b]$  
  a: Following axis  
  b: Leading axis  
  Synchronized position of leading axis b  
  Axes: geometry axis, channel axis, machine axis | RS         | R         | O 6       | S 1       |
| $\text{AA}_\text{EG\_ACTIVE}$ | BOOL     | $\text{AA}_\text{EG\_ACTIVE}[a,b]$  
  a: Following axis  
  b: Leading axis  
  Coupling for leading axis b is active, i.e. switched on  
  Axes: geometry axis, channel axis, machine axis | RS         | R         | O 6       | S 1       |

### 15.2.59 Safety Integrated

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync O S</th>
</tr>
</thead>
</table>
| $\text{A}_\text{STOPESI}$ | INT      | Current Safety Integrated Stop E for any axis:  
  Value 0: no Stop E  
  Value not equal to 0: There is currently a Stop E at one of the axes | RS         | R         | O 6       | S 4       |
| $\text{A}_\text{INSE}$    | BOOL     | $\text{A}_\text{INSE}[n]$  
  Image of a Safety input signal (ext. NCK interface)  
  n: Number of input 1 - ... | RS         | R         | O 6       | S 3       |
| $\text{A}_\text{INSED}$   | INT      | $\text{A}_\text{INSED}[n]$  
  Image of Safety input signals (ext. NCK interface)  
  n: Number of input word 1 - ... | RS         | R         | O 6       | S 3       |
| $\text{A}_\text{INSEP}$   | BOOL     | $\text{A}_\text{INSEP}[n]$  
  Image of a Safety input signal (ext. PLC interface)  
  n: Number of input 1 - ... | RS         | R         | O 6       | S 4       |
| $\text{A}_\text{INSEPD}$  | INT      | $\text{A}_\text{INSEPD}[n]$  
  Image of Safety input signals (ext. PLC interface)  
  n: Number of input word 0 - ... | RS         | R         | O 6       | S 3       |
| $\text{A}_\text{OUTSE}$   | BOOL     | $\text{A}_\text{OUTSE}[n]$  
  Image of a Safety output signal (ext. NCK interface)  
  n: Number of output 1 - ... | RS         | W         | O 6       | S 4       |
| $\text{A}_\text{OUTSED}$  | INT      | $\text{A}_\text{OUTSED}[n]$  
  Image of Safety output signals (ext. NCK interface)  
  n: Number of output word 1 - ... | RS         | W         | O 6       | S 3       |
| $\text{A}_\text{OUTSEP}$  | BOOL     | $\text{A}_\text{OUTSEP}[n]$  
  Image of a Safety output signal (ext. PLC interface)  
  n: Number of output 1 - ... | RS         | R         | O 6       | S 3       |
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
</table>
| $A\_OUTSEPD$ | INT    | $A\_OUTSEPD[n]$  
Image of Safety output signals (ext. PLC interface)  
n: Number of output word 0 - ... | RS        | R    |   |   |
| $A\_INSI$    | BOOL   | $A\_INSI[n]$  
Image of a Safety input signal (int. NCK interface)  
n: Number of input 1 - ... | RS        | R    |   | 4 |
| $A\_INSID$   | INT    | $A\_INSID[n]$  
Image of Safety input signals (int. NCK interface)  
n: Number of input word 1 - ... | RS        | R    |   | 3 |
| $A\_INSIP$   | BOOL   | $A\_INSIP[n]$  
Image of a Safety input signal (int. PLC interface)  
n: Number of input word 1 - ... | RS        | R    |   | 4 |
| $A\_INSIPD$  | INT    | $A\_INSIPD[n]$  
Image of Safety input signals (int. PLC interface)  
n: Number of input word 1 - ... | RS        | R    |   | 4 |
| $A\_OUTSI$   | BOOL   | $A\_OUTSI[n]$  
Image of a Safety output signal (int. NCK interface)  
n: Number of output 1 - ... | RS WS     | R W  |   | 4 |
| $A\_OUTSID$  | INT    | $A\_OUTSID[n]$  
Image of Safety output signals (int. NCK interface)  
n: Number of output word 1 - ... | RS WS     | R W  |   | 3 |
| $A\_OUTSIP$  | BOOL   | $A\_OUTSIP[n]$  
Image of a Safety output signal (int. PLC interface)  
n: Number of output 1 - ... | RS        | R    |   | 3 |
| $A\_OUTSIPD$ | INT    | $A\_OUTSIPD[n]$  
Image of Safety output signals (int. PLC interface)  
n: Number of output word 1 - ... | RS        | R    |   | 3 |
| $A\_MARKERSI$ | BOOL  | $A\_MARKERSI[n]$  
Markers for Safety programming  
n: Number of marker 1 - ... | RS WS     | R W  |   | 3 |
| $A\_MARKERSID$ | INT  | $A\_MARKERSID[n]$  
Marker word (32 bits) for Safety programming  
n: Number of marker word 1 - ... | RS WS     | R W  |   | 3 |
| $A\_MARKERSIP$ | BOOL | $A\_MARKERSIP[n]$  
Image of PLC Safety markers  
n: Number of marker 1 - ... | RS        | R    |   | 3 |
| $A\_MARKERSIPD$ | INT | $A\_MARKERSIPD[n]$  
Image of PLC Safety marker words  
n: Number of marker word 1 - ... | RS        | R    |   | 3 |
### List of System Variables

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Description: System variable/value range/index</th>
<th>Parts pr.</th>
<th>Sync</th>
<th>O</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_TIMERSI$</td>
<td>REAL</td>
<td>$A_TIMERSI[n]$, Safety timer - unit in seconds. Time is counted internally in multiples of the interpolation cycle; Counting for the time variable is started by assigning the value $A_TIMERSI[n]=&lt;\text{start value}&gt;$ To stop the counter variable, assign a negative value: $A_TIMERSI[n]=-1$ The current time can be read while the counter is active or stopped. Stopping the counter variable, by assigning -1 stops the last current time value which can then be read. n: Number of timer 1 - ...</td>
<td>RS WS R   W +</td>
<td>6</td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>$A_STATSID$</td>
<td>INT</td>
<td>$A_STATSID$, Safety: Status of cross-checking between NCK and PLC. If value is not equal to zero, there is a cross-checking error</td>
<td>RS R</td>
<td>6</td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>$A_CMDSI$</td>
<td>BOOL</td>
<td>$A_CMDSI[n]$, Safety: Control word for cross-checking between NCK and PLC. Array index n = 1: Increase timer for signal change monitoring to 10s n: Number of control signal for cross-checking NCK - PLC</td>
<td>RS WS R   W +</td>
<td>6</td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>$A_LEVELSID$</td>
<td>INT</td>
<td>$A_LEVELSID$, Safety: Display of signal change monitoring level. Indicates the current number of signals marked for cross-checking.</td>
<td>RS R</td>
<td>6</td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>$A_XFAULTSI$</td>
<td>INT</td>
<td>Information on Safety Integrated Stop F for an axis: Bit 0 is set: During crosschecking between NCK and 611D, an actual value error has been discovered on an axis. Bit 1 is set: During crosschecking between NCK and 611D, an error has been discovered on an axis and the wait time before Stop B is triggered is running or has expired ($MA_SAFE_STOP_SWITCH_TIME_F$).</td>
<td>RS R</td>
<td>6</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>$A_PLCSIIN$</td>
<td>BOOL</td>
<td>$A_PLCSIIN[n]$, Communication from PLC-SPL to NCK-SPL n: Number of signal 1 - ... from the PLC</td>
<td>RS R</td>
<td>6</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>$A_PLCSIOUT$</td>
<td>BOOL</td>
<td>$A_PLCSIOUT[n]$, Communication from NCK-SPL to PLC-SPL n: Number of signal 1 - ... to the PLC</td>
<td>RS WS R   W +</td>
<td>6</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Description: System variable/value range/index</td>
<td>Parts pr.</td>
<td>Sync</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$VA_{IS}$</td>
<td>REAL</td>
<td>$VA_{IS}[X]$ Reliable actual position (SISITEC) Axes: geometry axis, channel axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$VA_{STOPSI}$</td>
<td>INT</td>
<td>$VA_{STOPSI}[X]$ Current Safety Integrated Stop for the particular axis Value Meaning -1 No stop 0 Stop A 1 Stop B 2 Stop C 3 Stop D 4 Stop E 5 Stop F 10 Test stop NC 11 Test ext. pulse suppression Axes: geometry axis, channel axis</td>
<td>RS</td>
<td>R</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$VA_{XFAULTSI}$</td>
<td>INT</td>
<td>Information on Safety Integrated Stop F for this axis: Bit 0 is set: During crosschecking between NCK and 611D, an actual value error has been discovered. Bit 1 is set: During crosschecking between NCK and 611D, an error has been discovered and the wait time before Stop B ($MA_{SAFE_STOP_SWITCH_TIME_F}$) is triggered is running or has expired Axes: geometry axis, channel axis</td>
<td>R</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>
Appendix

A  Index .......................................................................................................................... A-702
B  Commands, Identifiers ............................................................................................. A-719
A Index

$A_CMDSI 15-698
$A_DAY 15-662
$A_DBB 15-657
$A_DBD 15-657
$A_DBR 15-657
$A_DBW 15-657
$A_DLB 15-658
$A_DLD 15-658
$A_DLR 15-658
$A_DLW 15-658
$A_DNO 15-646
$A_GG 15-647
$A_HOUR 15-662
$A_IN 15-657
$A_INA 15-657
$A_INCO 15-657
$A_INS 15-647
$A_LINK_TRANS_RATE 15-658
$A_MARKERSI 15-697
$A_MARKERSID 15-697
$A_MARKERSIP 15-697
$A_MARKERSIPD 15-697
$A_MINUTE 15-662
$A_MONIFACT 15-641
$A_MONTH 15-662
$A_MSECOND 15-662
$A_MYMILN 15-641
$A_MYMNLN 15-641
$A_OUT 15-657
$A_OUTA 15-657
$A_OUTSE 15-696
$A_OUTSED 15-696
$A_OUTSEP 15-696
$A_OUTSEPDE 15-697
$A_OUTSIE 15-697
$A_OUTSID 15-697
$A_OUTSIP 15-697
$A_OUTSIPD 15-697
$A_PBB_IN 15-658
$A_PBB_OUT 15-659
$A_PBD_IN 15-658
$A_PBD_OUT 15-659
$A_PBR_IN 15-658
$A_PBR_OUT 15-659
$A_PBW_IN 15-658
$A_PBW_OUT 15-659
$A_PLCSIIN 15-698
$A_PLCSIOUT 15-698
$A_PROBE 15-652
$A_PROT 15-655
$A_PROT 15-655
$A_SECOUND 15-662
$A_STATSID 15-698
$A_STOPESI 15-696
$A_TIMERSI 15-698
$A_TOOLMLN 15-641
$A_TOOLMN 15-641
$A_XFAULTSI 15-698
$A_YEAR 15-662
$AA_ACC 15-693
$AA_ACCLIWA 15-694
$AA_ACT_INDEX_AX_POS_NO 15-679
$AA_ALARM_STAT 15-695
$AA_COUP_ACT 9-361, 9-378, 13-503, 15-691
$AA_COUPT_OFFS 13-503, 15-693
$AA_CURR 15-687
$AA_DELT 15-685
$AA_DTB 15-684
$AA_DTBW 15-684
$AA_DTEB  15-684
$AA_DTEPB  15-685
$AA_DTEPW  15-685
$AA_DTEW  15-684
$AA_EG_ACTIVE  15-696
$AA_EG_AX  15-692
$AA_EG_DENOM  15-696
$AA_EG_NUM_LA  15-691
$AA_EG_NUMERA  15-695
$AA_EG_SYN  15-696
$AA_EG_SYNFA  15-691
$AA_EG_TYPE  15-695
$AA_ENC_ACTIVE  15-679
$AA_ENC_COMP  15-626
$AA_ENC_COMP_IS_MODULO  15-627
$AA_ENC_COMP_MAX  15-627
$AA_ENC_COMP_MIN  15-627
$AA_ENC_COMP_STEP  15-626
$AA_ENC1_ACTIVE  15-679
$AA_ENC2_ACTIVE  15-679
$AA_ESR_ENABLE  15-694
$AA_ESR_STAT  15-694
$AA_ESR_TRIGGER  15-695
$AA_ETRANS  15-681
$AA_FOC  15-691
$AA_FXS  15-690
$AA_IB  15-678
$AA_IBN  15-678
$AA_IEN  15-678
$AA_IM  15-678
$AA_IN_SYNC  15-693
$AA_IW  15-678
$AA_JERKLIMA  15-694
$AA_LEAD_P  15-692
$AA_LEAD_P_TURN  15-692
$AA_LEAD_SP  9-378, 15-692
$AA_LEAD_V  9-378, 15-692
$AA_LOAD  15-687
$AA_MASL_STAT  15-689
$AA_MEAACT  15-681
$AA_MEAS_P1_VALID  15-681
$AA_MEAS_P1  15-681
$AA_MEAS_P2_VALID  15-681
$AA_MEAS_P2  15-681
$AA_MEAS_P3_VALID  15-681
$AA_MEAS_P3  15-681
$AA_MEAS_P4_VALID  15-681
$AA_MEAS_POINT1  15-681
$AA_MEAS_POINT2  15-681
$AA_MEAS_POINT3  15-682
$AA_MEAS_POINT4  15-682
$AA_MEAS_SETANGLE  15-682
$AA_MEAS_SETPOINT  15-682
$AA_MEAS_SP_VALID  15-682
$AA_MM  15-680
$AA_MM1  15-680
$AA_MM2  15-680
$AA_MM3  15-680
$AA_MM4  15-681
$AA_MOTEND  5-230, 15-694
$AA_MW  15-680
$AA_MW1  15-680
$AA_MW2  15-680
$AA_MW3  15-680
$AA_MW4  15-680
$AA_OFF  15-682
$AA_OFF_LIMIT  15-682
$AA_OFF_VAL  15-682
$AA_OSCILL_REVERSE_POS1  15-685
$AA_OSCILL_REVERSE_POS2  15-685
$AA_OVR  15-686
$AA_POLFA  15-695
$AA_POLFA_VALID  15-695
$AA_POWER  15-687
$AA_PROG_INDEX_AX_POS_NO  15-679
$AA_QEC  15-627
$AA_QEC_ACCEL_1  15-627
$AA_QEC_ACCEL_2  15-628
$AA_QEC_ACCEL_3  15-628
$AA_QEC_COARSE_STEPS  15-627
$AA_QEC_DIRECTIONAL  15-628
$AA_QEC_FINE_STEPS  15-627
$AA_QEC_LEARNING_RATE  15-628
$AA_QEC_MEAS_TIME_1  15-628
$AA_QEC_MEAS_TIME_2  15-628
$AA_QEC_MEAS_TIME_3  15-628
$AA_QEC_TIME_1  15-628
$AA_QEC_TIME_2  15-628
$AA_QEC_TIME_3  15-628
$AA_REF  15-689
$AA_REPOS_DELAY  15-678
$AA_S 15-667
$AA_SCPAR 5-232, 15-694
$AA_SCTRACE 15-693
$AA_SNLAX_STAT 15-689
$AA_SOFTENDN 15-683
$AA_SOFTENDP 15-683
$AA_STAT 15-688
$AA_SYNC 15-693
$AA_TOFF 15-683
$AA_TOFF_LIMIT 15-683
$AA_TOFF_PREP_DIFF 15-683
$AA_TOFF_VAL 15-683
$AA_TORQUE 15-687
$AA_TYP 15-689
$AA_VACTB 15-686
$AA_VACTM 15-687
$AA_VACTW 15-686
$AA_VC 15-686
$AA_VELOLIMA 15-694
$AC_ACTUAL_PARTS 15-672
$AC_ALARM_STAT 15-672
$AC_ASUB 15-654
$AC_AXCTSWA 15-695
$AC_BLOCKTYPE 15-651
$AC_CONSTCUT_S 15-667
$AC_CUTTING_TIME 15-672
$AC_CYCLE_TIME 15-672
$AC_DELT 15-665
$AC_DRF 15-681
$AC_DTBB 15-664
$AC_DTBW 15-663
$AC_DTEB 15-664
$AC_DTEW 15-664
$AC_ESR_TRIGGER 15-672
$AC_F 15-665
$AC_FCT0 15-671
$AC_FCT1 15-671
$AC_FCT1C 15-670
$AC_FCT1LL 15-670
$AC_FCT1UL 15-670
$AC_FCT2 15-671
$AC_FCT2C 15-670
$AC_FCT2LL 15-670
$AC_FCT2UL 15-670
$AC_FCT3 15-671
$AC_FCT3C 15-670
$AC_FCT3LL 15-670
$AC_FCT3UL 15-670
$AC_FCTUL 15-670
$AC_FIFO1 15-656
$AC_FIFO2 15-656
$AC_FIFO3 15-656
$AC_GOMODE 15-673
$AC_IPO_BUF 15-651
$AC_IW_STAT 15-651
$AC_IW_TU 15-651
$AC_JOG_COORD 15-652
$AC_LIFTFAST 15-652
$AC_MARKER 15-591
$AC_MEA 15-652
$AC_MEAS_ACT_PLAN 15-674
$AC_MEAS_CHBFR 15-674
$AC_MEAS_CHSFR 15-674
$AC_MEAS_CORNER_ANGLE 15-676
$AC_MEAS_CORNER_SETANGLE 15-674
$AC_MEAS_D_NUMBER 15-675
$AC_MEAS_DIAMETER 15-676
$AC_MEAS_DIR_APPROACH 15-674
$AC_MEAS_FINE_TRANS 15-674
$AC_MEAS_FRAME 15-676
$AC_MEAS_FRAME_SELECT 15-674
$AC_MEAS_LATCH 15-673
$AC_MEAS_NCBFR 15-674
$AC_MEAS_P1_COORD 15-673
$AC_MEAS_P2_COORD 15-673
$AC_MEAS_P3_COORD 15-673
$AC_MEAS_P4_COORD 15-673
$AC_MEAS_PFRA 15-675
$AC_MEAS_PFRAME 15-675
$AC_MEAS_RESULTS 15-677
$AC_MEAS_SCALEUNIT 15-677
$AC_MEAS_SEMA 15-673
$AC_MEAS_SET_COORD 15-673
$AC_MEAS_T_NUMBER 15-675
$AC_MEAS_TOOL_LENGTH 15-676
$AC_MEAS_TOOL_MASK 15-675
$AC_MEAS_TYPE 15-675
$AC_MEAS_UIFR 15-674
$AC_MEAS_VALID 15-676
$AC_MEAS_WP_ANGLE 15-676
$AC_MEAS_WP_SETANGLE 15-674
$AC_MONMIN 15-646
$AC_MSNUM 15-669
$AC_MTHNUM 15-670
$AC_OPERATING_TIME 15-672
$AC_OVR 15-665
$AC_PARAM 15-591
$AC_PATHACC 15-666
$AC_PATHJERK 15-666
$AC_PATHN 15-663
$AC_PLTBB 15-664
$AC_PLTEB 15-664
$AC_PRESET 15-681
$AC_PROG 15-651
$AC_PRTIME_A 15-663
$AC_PRTIME_A_INC 15-663
$AC_PRTIME_M 15-663
$AC_PRTIME_M_INC 15-663
$AC_REPOS_PATH_MODE 15-664
$AC_REQUIRED_PARTS 15-672
$AC_RETPONT 15-683
$AC_ROT_SYS 15-652
$AC_SDIR 15-667
$AC_SERUPRO 15-677
$AC_SGEAR 15-668
$AC_SMODE 15-668
$AC_SPECIAL_PARTS 15-672
$AC_STAT 15-651
$AC_SYNA_MEM 15-651
$AC_SYSTEM_MARKER 15-592
$AC_SYSTEM_PARAM 15-591
$AC_TANEB 15-651
$AC_TC 15-640
$AC_TC_ACKC 15-660
$AC_TC_ACKT 15-659
$AC_TC_CMDC 15-660
$AC_TC_CMDT 15-659
$AC_TC_FCT 15-660
$AC_TC_LFN 15-660
$AC_TC_LFO 15-661
$AC_TC_LMYN 15-660
$AC_TC_LTN 15-661
$AC_TC_LTO 15-661
$AC_TC_MFO 15-661
$AC_TC_MMYN 15-660
$AC_TC_MTN 15-661
$AC_TC_MTO 15-661
$AC_TC_STATUS 15-660
$AC_TC_THNO 15-660
$AC_TC_TNO 15-660
$AC_TIME 15-662
$AC_TIMEC 15-662
$AC_TIMER 15-662
$AC_TOOLO_ACT 15-639
$AC_TOOLO_DIFF 15-639
$AC_TOOLO_END 15-639
$AC_VACTB 15-666
$AC_VACTW 15-666
$AC_VC 15-666
$AN_AXCTAS 15-695
$AN_AXCTSWA 15-695
$AN_BUS_FAIL_TRIGGER 15-672
$AN_CEC 15-629
$AN_CEC_DIRECTION 15-629
$AN_CEC_MAX 15-629
$AN_CEC_MIN 15-629
$AN_CEC_MULT_BY_TABLE 15-629
$AN_CEC_OUTPUT_AXIS 15-629
$AN_CEC_STEP 15-629
$AN_CEC_DELAY 15-629
$AN_CEC_INPUT_AXIS 15-629
$AN_CEC_IS_MODULO 15-629
$AN_CEC_VERSION 15-629
$SC_A 15-631
$SC_A_PROG 15-633
$SC_ALL_PROG 15-634
$SC_B 15-631
$SC_B_PROG 15-633
<table>
<thead>
<tr>
<th>Appendix 11.02 Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_DL$ 15-632</td>
</tr>
<tr>
<td>$C_DL_PROG$ 15-633</td>
</tr>
<tr>
<td>$C_H$ 15-631</td>
</tr>
<tr>
<td>$C_I$ 15-632</td>
</tr>
<tr>
<td>$C_I_NUM$ 15-634</td>
</tr>
<tr>
<td>$C_I_ORDER$ 15-635</td>
</tr>
<tr>
<td>$C_IN$ 15-659</td>
</tr>
<tr>
<td>$C_INC_PROG$ 15-634</td>
</tr>
<tr>
<td>$C_J$ 15-632</td>
</tr>
<tr>
<td>$C_J_NUM$ 15-634</td>
</tr>
<tr>
<td>$C_J_ORDER$ 15-635</td>
</tr>
<tr>
<td>$C_K$ 15-632</td>
</tr>
<tr>
<td>$C_K_NUM$ 15-634</td>
</tr>
<tr>
<td>$C_K_ORDER$ 15-635</td>
</tr>
<tr>
<td>$C_L$ 15-632</td>
</tr>
<tr>
<td>$C_M$ 15-632</td>
</tr>
<tr>
<td>$C_MACPAR$ 15-635</td>
</tr>
<tr>
<td>$C_ME$ 15-635</td>
</tr>
<tr>
<td>$C_OUT$ 15-659</td>
</tr>
<tr>
<td>$C_TE$ 15-635</td>
</tr>
<tr>
<td>$C_TS$ 15-632</td>
</tr>
<tr>
<td>$C_TS_PROG$ 15-634</td>
</tr>
<tr>
<td>$C_TYP_PROG$ 15-634</td>
</tr>
<tr>
<td>$C_Z$ 15-632</td>
</tr>
<tr>
<td>$C_Z_PROG$ 15-633</td>
</tr>
<tr>
<td>$MC_COMPRESS_VELO_TOL. 9-384</td>
</tr>
<tr>
<td>$P_ACTFBFRAME$ 15-637</td>
</tr>
<tr>
<td>$P_CONSTCUT_S$ 15-667</td>
</tr>
<tr>
<td>$P_COUP_OFFS$ 15-693</td>
</tr>
<tr>
<td>$P_CTABDEF$ 15-648</td>
</tr>
<tr>
<td>$P_CYCFR$ 15-592</td>
</tr>
<tr>
<td>$P_CYCFRAME$ 15-636</td>
</tr>
<tr>
<td>$P_D$ 15-641</td>
</tr>
<tr>
<td>$P_DLNO$ 15-638</td>
</tr>
<tr>
<td>$P_DRYRUN$ 15-649</td>
</tr>
<tr>
<td>$P_EG_BC$ 15-691</td>
</tr>
<tr>
<td>$P_EP$ 15-677</td>
</tr>
<tr>
<td>$P_EPM$ 15-678</td>
</tr>
<tr>
<td>$P_EXTFR$ 15-592</td>
</tr>
<tr>
<td>$P_EXTFRAME$ 15-636</td>
</tr>
<tr>
<td>$P_EXTGG$ 15-647</td>
</tr>
<tr>
<td>$P_F$ 15-665</td>
</tr>
<tr>
<td>$P_FA$ 15-685</td>
</tr>
<tr>
<td>$P_GG$ 15-647</td>
</tr>
<tr>
<td>$P_GWPS$ 15-670</td>
</tr>
<tr>
<td>$P_H$ 15-641</td>
</tr>
<tr>
<td>$P_IFRAME$ 15-637</td>
</tr>
<tr>
<td>$P_ISTEST$ 15-655</td>
</tr>
<tr>
<td>$P_LIFTFAST$ 15-653</td>
</tr>
<tr>
<td>$P_MAG$ 15-643</td>
</tr>
<tr>
<td>$P_MAGA$ 15-645</td>
</tr>
<tr>
<td>$P_MAGDISL$ 15-644</td>
</tr>
<tr>
<td>$P_MAGDISS$ 15-643</td>
</tr>
<tr>
<td>$P_MAGHLT$ 15-645</td>
</tr>
<tr>
<td>$P_MAGM$ 15-643</td>
</tr>
<tr>
<td>$P_MAGNA$ 15-645</td>
</tr>
<tr>
<td>$P_MAGDIS$ 15-643</td>
</tr>
<tr>
<td>$P_MAGNS$ 15-644</td>
</tr>
<tr>
<td>$P_MAGREL$ 15-644</td>
</tr>
<tr>
<td>$P_MAGREL$ 15-644</td>
</tr>
<tr>
<td>$P_MAGN$ 15-644</td>
</tr>
<tr>
<td>$P_MAGN$ 15-644</td>
</tr>
<tr>
<td>$P_MSNUM$ 15-669</td>
</tr>
<tr>
<td>$P_MTHNUM$ 15-669</td>
</tr>
<tr>
<td>$P_MTHSDC$ 15-645</td>
</tr>
<tr>
<td>$P_NCBFR$ 15-593</td>
</tr>
<tr>
<td>$P_NCBFRAME$ 15-637</td>
</tr>
</tbody>
</table>

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
<table>
<thead>
<tr>
<th>Variable</th>
<th>Page Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{NCBFRMASK}$</td>
<td>15-637</td>
</tr>
<tr>
<td>$P_{NUM_SPINDLES}$</td>
<td>15-669</td>
</tr>
<tr>
<td>$P_{OFFN}$</td>
<td>15-649</td>
</tr>
<tr>
<td>$P_{PARTFR}$</td>
<td>15-592</td>
</tr>
<tr>
<td>$P_{PARTFRAME}$</td>
<td>15-636</td>
</tr>
<tr>
<td>$P_{PFRAFME}$</td>
<td>15-637</td>
</tr>
<tr>
<td>$P_{POLF}$</td>
<td>15-678</td>
</tr>
<tr>
<td>$P_{POLF_VALID}$</td>
<td>15-678</td>
</tr>
<tr>
<td>$P_{PROG}$</td>
<td>15-649</td>
</tr>
<tr>
<td>$P_{PROG_EVENT}$</td>
<td>15-649</td>
</tr>
<tr>
<td>$P_{PROGPATH}$</td>
<td>15-649</td>
</tr>
<tr>
<td>$P_{REPINF}$</td>
<td>15-648</td>
</tr>
<tr>
<td>$P_{S}$</td>
<td>15-667</td>
</tr>
<tr>
<td>$P_{SAUTOGEAR}$</td>
<td>15-668</td>
</tr>
<tr>
<td>$P_{SDIR}$</td>
<td>15-667</td>
</tr>
<tr>
<td>$P_{SEARCH}$</td>
<td>15-647</td>
</tr>
<tr>
<td>$P_{SEARCH_MASLC}$</td>
<td>15-689</td>
</tr>
<tr>
<td>$P_{SEARCH_MASLD}$</td>
<td>15-690</td>
</tr>
<tr>
<td>$P_{SEARCH_S}$</td>
<td>15-667</td>
</tr>
<tr>
<td>$P_{SEARCH_SDIR}$</td>
<td>15-667</td>
</tr>
<tr>
<td>$P_{SEARCH_SGEAR}$</td>
<td>15-669</td>
</tr>
<tr>
<td>$P_{SEARCH_SPO$</td>
<td>15-669</td>
</tr>
<tr>
<td>$P_{SEARCH_SPO$</td>
<td>15-669</td>
</tr>
<tr>
<td>$P_{SEARCH1}$</td>
<td>15-647</td>
</tr>
<tr>
<td>$P_{SEARCH2}$</td>
<td>15-647</td>
</tr>
<tr>
<td>$P_{SEARCHL}$</td>
<td>15-648</td>
</tr>
<tr>
<td>$P_{SETFR}$</td>
<td>15-592</td>
</tr>
<tr>
<td>$P_{SETFRAME}$</td>
<td>15-636</td>
</tr>
<tr>
<td>$P_{SGEAR}$</td>
<td>15-668</td>
</tr>
<tr>
<td>$P_{SIM}$</td>
<td>15-648</td>
</tr>
<tr>
<td>$P_{SMODE}$</td>
<td>15-668</td>
</tr>
<tr>
<td>$P_{STACK}$</td>
<td>15-649</td>
</tr>
<tr>
<td>$P_{SUBPAR}$</td>
<td>15-648</td>
</tr>
<tr>
<td>$P_{TC}$</td>
<td>15-640</td>
</tr>
<tr>
<td>$P_{TCANG}$</td>
<td>15-640</td>
</tr>
<tr>
<td>$P_{TCDIFF}$</td>
<td>15-640</td>
</tr>
<tr>
<td>$P_{TCSTAT}$</td>
<td>15-640</td>
</tr>
<tr>
<td>$P_{TOOL}$</td>
<td>15-638</td>
</tr>
<tr>
<td>$P_{TOOLD}$</td>
<td>15-642</td>
</tr>
<tr>
<td>$P_{TOOLEN}$</td>
<td>15-646</td>
</tr>
<tr>
<td>$P_{TOOLENV}$</td>
<td>15-646</td>
</tr>
<tr>
<td>$P_{TOOLENVN}$</td>
<td>15-646</td>
</tr>
<tr>
<td>$P_{TOOLEXIST}$</td>
<td>15-641</td>
</tr>
<tr>
<td>$P_{TOOLT}$</td>
<td>15-642</td>
</tr>
<tr>
<td>$P_{TOOLO}$</td>
<td>15-639</td>
</tr>
<tr>
<td>$P_{TOOLP}$</td>
<td>15-639</td>
</tr>
<tr>
<td>$P_{TOOLL}$</td>
<td>15-642</td>
</tr>
<tr>
<td>$P_{TOOLND}$</td>
<td>15-641</td>
</tr>
<tr>
<td>$P_{TOOLNDL}$</td>
<td>15-642</td>
</tr>
<tr>
<td>$P_{TOOLNO}$</td>
<td>15-639</td>
</tr>
<tr>
<td>$P_{TOOLNT}$</td>
<td>15-642</td>
</tr>
<tr>
<td>$P_{TRAFO}$</td>
<td>15-652</td>
</tr>
<tr>
<td>$P_{TRAFO_PAR}$</td>
<td>15-652</td>
</tr>
<tr>
<td>$P_{TRAFO_PARSET}$</td>
<td>15-652</td>
</tr>
<tr>
<td>$P_{TRAFR}$</td>
<td>15-593</td>
</tr>
<tr>
<td>$P_{UBFR}$</td>
<td>15-636</td>
</tr>
<tr>
<td>$P_{UIFR}$</td>
<td>15-592</td>
</tr>
<tr>
<td>$P_{UIFRNUM}$</td>
<td>15-637</td>
</tr>
<tr>
<td>$P_{USEKT}$</td>
<td>15-642</td>
</tr>
<tr>
<td>$P_{VDITCP}$</td>
<td>15-646</td>
</tr>
<tr>
<td>$P_{WPFR}$</td>
<td>15-592</td>
</tr>
<tr>
<td>$P_{WPFRAME}$</td>
<td>15-636</td>
</tr>
<tr>
<td>$PA_{ACCLIMA}$</td>
<td>15-694</td>
</tr>
<tr>
<td>$PA_{JERKLIMA}$</td>
<td>15-694</td>
</tr>
<tr>
<td>$PA_{VELOLIMA}$</td>
<td>15-694</td>
</tr>
<tr>
<td>$PI$</td>
<td>15-649</td>
</tr>
<tr>
<td>$SA_LEAD_TYPE$</td>
<td>9-377, 9-378</td>
</tr>
<tr>
<td>$SC_PA_ACTIV_IMM$</td>
<td>15-601</td>
</tr>
<tr>
<td>$SC_PA_CEN$</td>
<td>15-603</td>
</tr>
<tr>
<td>$SC_PA_CENT_ORD$</td>
<td>15-603</td>
</tr>
<tr>
<td>$SC_PA_CONT_ABS$</td>
<td>15-603</td>
</tr>
<tr>
<td>$SC_PA_CONT_NUM$</td>
<td>15-602</td>
</tr>
<tr>
<td>$SC_PA_CONT_TYP$</td>
<td>15-602</td>
</tr>
<tr>
<td>$SC_PA_LIM_3DIM$</td>
<td>15-602</td>
</tr>
<tr>
<td>$SC_PA_MINUS_LIM$</td>
<td>15-602</td>
</tr>
<tr>
<td>$SC_PA_ORI$</td>
<td>15-602</td>
</tr>
<tr>
<td>$SC_PA_PLUS_LIM$</td>
<td>15-602</td>
</tr>
<tr>
<td>$SC_PA_T_W$</td>
<td>15-601</td>
</tr>
<tr>
<td>$SN_PA_ACTIV_IM$</td>
<td>15-630</td>
</tr>
<tr>
<td>$SN_PA_CENT_ABS$</td>
<td>15-631</td>
</tr>
<tr>
<td>$SN_PA_CENT_ORD$</td>
<td>15-631</td>
</tr>
<tr>
<td>$SN_PA_CONT_ABS$</td>
<td>15-631</td>
</tr>
<tr>
<td>$SN_PA_CONT_NUM$</td>
<td>15-630</td>
</tr>
<tr>
<td>$SSN_{PA_CONT_ORD}$ 15-631</td>
<td>$STC_{CARR37}$ 15-601</td>
</tr>
<tr>
<td>$SSN_{PA_CONT_TYP}$ 15-631</td>
<td>$STC_{CARR38}$ 15-601</td>
</tr>
<tr>
<td>$SSN_{PA_LIM_3DIM}$ 15-630</td>
<td>$STC_{CARR39}$ 15-601</td>
</tr>
<tr>
<td>$SSN_{PA_MINUS_LIM}$ 15-630</td>
<td>$STC_{CARR4}$ 15-594</td>
</tr>
<tr>
<td>$SSN_{PA_ORI}$ 15-630</td>
<td>$STC_{CARR40}$ 15-601</td>
</tr>
<tr>
<td>$SSN_{PA_PLUS_LIM}$ 15-630</td>
<td>$STC_{CARR5}$ 15-594</td>
</tr>
<tr>
<td>$SSN_{PA_T_W}$ 15-630</td>
<td>$STC_{CARR6}$ 15-594</td>
</tr>
<tr>
<td>$STC_{ADPT1}$ 15-626</td>
<td>$STC_{CARR7}$ 15-594</td>
</tr>
<tr>
<td>$STC_{ADPT2}$ 15-626</td>
<td>$STC_{CARR8}$ 15-594</td>
</tr>
<tr>
<td>$STC_{ADPT3}$ 15-626</td>
<td>$STC_{CARR9}$ 15-594</td>
</tr>
<tr>
<td>$STC_{ADPTT}$ 15-626</td>
<td>$STC_{DP1}$ 15-603</td>
</tr>
<tr>
<td>$STC_{CARR1}$ 15-593</td>
<td>$STC_{DP10}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR1...14}$ 8-346</td>
<td>$STC_{DP11}$ 15-605</td>
</tr>
<tr>
<td>$STC_{CARR10}$ 15-594</td>
<td>$STC_{DP12}$ 15-605</td>
</tr>
<tr>
<td>$STC_{CARR11}$ 15-594</td>
<td>$STC_{DP13}$ 15-605</td>
</tr>
<tr>
<td>$STC_{CARR12}$ 15-595</td>
<td>$STC_{DP14}$ 15-605</td>
</tr>
<tr>
<td>$STC_{CARR13}$ 15-595</td>
<td>$STC_{DP15}$ 15-605</td>
</tr>
<tr>
<td>$STC_{CARR14}$ 15-595</td>
<td>$STC_{DP16}$ 15-605</td>
</tr>
<tr>
<td>$STC_{CARR15}$ 15-595</td>
<td>$STC_{DP17}$ 15-606</td>
</tr>
<tr>
<td>$STC_{CARR16}$ 15-595</td>
<td>$STC_{DP18}$ 15-606</td>
</tr>
<tr>
<td>$STC_{CARR17}$ 15-595</td>
<td>$STC_{DP19}$ 15-606</td>
</tr>
<tr>
<td>$STC_{CARR18}$ 15-595</td>
<td>$STC_{DP2}$ 15-603</td>
</tr>
<tr>
<td>$STC_{CARR18[m]}$ 8-346</td>
<td>$STC_{DP20}$ 15-606</td>
</tr>
<tr>
<td>$STC_{CARR19}$ 15-595</td>
<td>$STC_{DP21}$ 15-606</td>
</tr>
<tr>
<td>$STC_{CARR2}$ 15-593</td>
<td>$STC_{DP22}$ 15-606</td>
</tr>
<tr>
<td>$STC_{CARR20}$ 15-595</td>
<td>$STC_{DP23}$ 15-607</td>
</tr>
<tr>
<td>$STC_{CARR21}$ 15-596</td>
<td>$STC_{DP24}$ 15-607</td>
</tr>
<tr>
<td>$STC_{CARR22}$ 15-596</td>
<td>$STC_{DP25}$ 15-607</td>
</tr>
<tr>
<td>$STC_{CARR23}$ 15-596</td>
<td>$STC_{DP3}$ 15-603</td>
</tr>
<tr>
<td>$STC_{CARR24}$ 15-596</td>
<td>$STC_{DP4}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR24[m]}$ 8-348</td>
<td>$STC_{DP5}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR25}$ 15-596</td>
<td>$STC_{DP6}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR26}$ 15-596</td>
<td>$STC_{DP7}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR27}$ 15-597</td>
<td>$STC_{DP8}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR28}$ 15-597</td>
<td>$STC_{DP9}$ 15-604</td>
</tr>
<tr>
<td>$STC_{CARR29}$ 15-598</td>
<td>$STC_{DPC1}$ 15-609</td>
</tr>
<tr>
<td>$STC_{CARR3}$ 15-594</td>
<td>$STC_{DPC10}$ 15-609</td>
</tr>
<tr>
<td>$STC_{CARR30}$ 15-598</td>
<td>$STC_{DPC2}$ 15-609</td>
</tr>
<tr>
<td>$STC_{CARR31}$ 15-598</td>
<td>$STC_{DPC3}$ 15-609</td>
</tr>
<tr>
<td>$STC_{CARR32}$ 15-599</td>
<td>$STC_{DPC1}$ 15-609</td>
</tr>
<tr>
<td>$STC_{CARR33}$ 15-599</td>
<td>$STC_{DPCS1}$ 15-610</td>
</tr>
<tr>
<td>$STC_{CARR34}$ 15-600</td>
<td>$STC_{DPCS10}$ 15-610</td>
</tr>
<tr>
<td>$STC_{CARR35}$ 15-600</td>
<td>$STC_{DPCS2}$ 15-610</td>
</tr>
<tr>
<td>$STC_{CARR36}$ 15-600</td>
<td>$STC_{DPCS3}$ 15-610</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
</tr>
<tr>
<td>$TC_SC54</td>
<td>15-613</td>
</tr>
<tr>
<td>$TC_SC61</td>
<td>15-613</td>
</tr>
<tr>
<td>$TC_SC63</td>
<td>15-613</td>
</tr>
<tr>
<td>$TC_SC64</td>
<td>15-613</td>
</tr>
<tr>
<td>$TC_SC71</td>
<td>15-613</td>
</tr>
<tr>
<td>$TC_TP1</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP10</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP11</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP2</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP3</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP4</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP5</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP6</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP7</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP8</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TP9</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TPC1</td>
<td>15-619</td>
</tr>
<tr>
<td>$TC_TPC10</td>
<td>15-620</td>
</tr>
<tr>
<td>$TC_TPC2</td>
<td>15-620</td>
</tr>
<tr>
<td>$TC_TPCS1</td>
<td>15-620</td>
</tr>
<tr>
<td>$TC_TPCS10</td>
<td>15-620</td>
</tr>
<tr>
<td>$TC_TPCS2</td>
<td>15-620</td>
</tr>
<tr>
<td>$TC_TPG1</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG2</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG3</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG4</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG5</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG6</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG7</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG8</td>
<td>15-621</td>
</tr>
<tr>
<td>$TC_TPG9</td>
<td>15-621</td>
</tr>
<tr>
<td>$VA_COUP_OFFS</td>
<td>15-693</td>
</tr>
<tr>
<td>$VA_CURR</td>
<td>15-688</td>
</tr>
<tr>
<td>$VA_DIST_TORQ</td>
<td>15-688</td>
</tr>
<tr>
<td>$VA_DP_ACT_TEL</td>
<td>15-688</td>
</tr>
<tr>
<td>$VA_DPE</td>
<td>15-693</td>
</tr>
<tr>
<td>$VA_EG_SYNCDIFF</td>
<td>15-691</td>
</tr>
<tr>
<td>$VA_EG_SYNCDIFF_S</td>
<td>15-692</td>
</tr>
<tr>
<td>$VA_FOC</td>
<td>15-691</td>
</tr>
<tr>
<td>$VA_FXS</td>
<td>15-690</td>
</tr>
<tr>
<td>$VA_FXS_INFO</td>
<td>15-690</td>
</tr>
<tr>
<td>$VA_IM</td>
<td>15-679</td>
</tr>
<tr>
<td>$VA_IM1</td>
<td>15-680</td>
</tr>
<tr>
<td>$VA_IM2</td>
<td>15-680</td>
</tr>
</tbody>
</table>
## Index

### A

- Actual value and setpoint coupling 9-376
- Actual-value coupling 13-495
- Adaptive control, additive 10-424
- Adaptive control, multiplicative 10-425
- Angle of rotation $\alpha_1$, $\alpha_2$ 8-346
- Angle offset/angle increment of the rotary axes 8-348
- Angle reference 13-501
- Approaching coded positions 5-186
- Arithmetic functions 1-46
- Arithmetic operations/functions 1-46
- Arithmetic parameter 1-26
- Array definition 1-34
- Array definition, value lists 1-36
- Array index 1-35
- Assign and start interrupt routine 1-79
- Assignments 1-45
- ASUB 10-452
- Asynchronized oscillation 11-456
- Automatic "GET" 1-87
- Automatic path segmentation 12-480
- Auxiliary functions 10-415
- Auxiliary functions 12-480
- Axial feed 10-433
- Axial leading value coupling 9-375
- Axis
  - Container 13-526
  - Local 13-526
- Axis container 13-526, 13-528
- Axis coordination 10-434
- Axis functions 13-489
- Axis transfer
  - Release axis 1-86
- Axis transfer
  - GET 1-85
  - Get axis 1-86
  - RELEASE 1-85

### B

- Backlash 13-493
- Block display 2-121, 2-125
- Block search 10-452

### C

- Calculate circle data 14-557
- Calculate intersection of two contour elements 14-542
- Calling frame 6-244
- Calling up a program in ISO language indirectly with ISOCALL 2-121
- CANCEL 10-453
- Cancel synchronized action 10-449
- CASE instruction 1-65
- Channel-specific frames 6-257
- CHECKSUM 1-98
- Circular interpolation 5-212
- Circumferential milling 8-328
- Clamping axis/spindle 13-526
- Clearance control 10-426
- Coarse offset 6-248
- Command axes 10-430
- Command elements 10-397
- Comparison and logic operators 1-48
  - Priority of operators 1-53
- Compressor 5-196, 5-211
- Compressor for orientations COMPON, COMPCURV 5-197
- Computing capacity 13-522
- Concatenation of strings 1-58
- Constraints for transformations 7-302
- Contour element 14-546, 14-548
- Contour elements, intersection 14-554
- Contour preparation
  - Relief cut elements 14-544
- Contour preparation 14-543, 14-550
- Contour table 14-543, 14-550
- Control structures 1-67
- Coupled motion 9-358
  - Coupled-motion axes 9-359
  - Coupling factor 9-360
  - Coupled-axis combinations 9-359
Coupled-axis motion 10-438
Coupling  9-352, 9-358
Coupling  13-495
Cov.com, user cycles 2-138
Create interrupt routine as subprogram 1-78
CS  9-352
CTAB  9-369
CTABDEF  9-365
CTABEND  9-365
CTABINV  9-369
Current
  Angular offset 13-503
  Coupling status following spindle 13-503
Current block display 2-125
Current channel basic frames 6-259
Current first basic frame in the channel 6-260
Current NCU-global basic frames 6-259
Current programmable frame 6-262
Current settable frame 6-261
Current system frames 6-259, 6-261, 6-262
Current total frame 6-262
Curve parameter 5-211
Curve tables 9-362
CUT  14-555
Cutter
  Reference point (FH) 8-334
  Tip (FS) 8-334
Cutting edge number 8-341
Cycles
  Setting parameters for user cycles 2-136, 2-138
Cylinder peripheral curve transformation 7-290, 7-294
  Offset contour normal OFFN 7-292

D
D numbers
  Check 8-342
  Determine T number 8-344
  Free assignment 8-341
  Rename 8-343
DC link backup 13-519
Deactivate/reactivate interrupt routine 1-79
Deactivating frames 6-252
Deactivation position 13-501
Defining user data 3-156
Degrees 9-364
DELETE 1-93
Delete couplings 13-502
Delete distance-to-go 5-221
Delete distance-to-go with preparation 10-418
Deletion of distance-to-go 10-418, 11-462
Denominator polynomial 5-209
Deselect transformation: TRAFOOF 7-304
Direct axis transfer: GETD 1-87
Displaying the block number programmed last 2-121
DRF offset 6-249
Drive-independent reactions 13-514
Drive-independent retract 13-521
Drive-independent stop 13-520
Dwell time 1-76

E
EG
  Electronic gear 13-505
EGONSYNE 13-508
Electronic gear 13-505
End of program 10-451
Endless program 1-70
Error checkback 14-543, 14-550
Error responses 10-442
Euler angle 8-336
Evaluation function 10-423
EXECTAB 14-542
EXECUTE 4-176
Executing external subprogram 2-132
EXTCALL 2-132
Extended measuring function 5-218, 7-279
Extended stopping and retract 13-513
External zero offset 6-250
F

- F word polynomial  5-212
- Face milling  7-272
- Face turning
  - External machining  14-543
  - Internal machining  14-543
- FAxis 9-352, 9-358, 9-364, 9-375
- Feed
  - Axial 10-433
- FGROUP
  - Axes 5-211
- FIFO variable 10-413
- Fine offset 6-248
- First basic frame in the channel 6-258
- Flag variables 10-409
- Following axis 9-375
- FOR 1-68
- Frame calculation 6-253
- Frame chaining 6-245, 6-263
- Frame rotation definition 6-247
- Frame variable
  - Coordinate transformation call 6-236
- Frame variables 6-236
  - Assigning values 6-241
  - Definition of new frames 6-247
  - Predefined frame variables 6-237
  - Reading or changing frame components 6-243
- Friction 13-493

G

- G code 5-211
  - Group 5-213
- G643 5-212
- Generator operation 13-519
- GUD
  - Automatic activation 3-162

H

- Hold time 11-459
Learn compensation characteristics  13-493
Linear interpolation  5-211, 5-212
Link axis  13-526
Link communication  13-522
Link module  13-523
Link variable
  Global  13-523
Logic operators  1-51
Longitudinal turning
  External machining  14-543
  Internal machining  14-543
Lower/upper case  1-59

M
M commands  12-479
M function
  Three-digit  2-143
M6
    Subprogram call  2-136
MAC
  Automatic activation  3-162
MACH  14-543
Machine
  State, global  13-523
Machine and setting data  10-412
Macro technology  12-479
Macros  2-142
Max/min indicator  14-546, 14-548
MEAFRAME  6-253, 6-256
Measured value recording  5-217
Measurement  10-441
Measurement results  5-221
Measurements with touch trigger probe
  Programming measuring blocks  5-216
  Status variable  5-216
Measuring probe status  5-222
Memory
  Memory structure  3-146
  Program memory  3-146
  User memory  3-146
Minimum position/maximum position of the rotary axis
  8-348
Mode  11-463
Mode change  10-450
Motion control  13-534
Motion-synchronized actions
  Actions  10-402
  Overview  10-404
Motion-synchronous actions
  Programming  10-395

N
N 9-364
NC Stop  10-451
NCU
  Link  13-523
  NCU-global basic frames  6-256
  NCU-global settable frames  6-257
  NCU-to-NCU communication  13-523
  Nesting depth  1-69
  Networked NCUs  13-523
  NEWCONF  1-90
  Nibbling  12-476
  Nibbling on  12-476

O
OEM addresses  5-228
OEM functions  5-228
OEM interpolations  5-228
Offset contour normal OFFN  7-292
Offset of the rotary axes  8-348
Online tool length compensation  7-284
Online tool offset  10-428
Operating mode  5-220
Orientation axes  7-269, 7-274, 7-276
Oscillating axis  11-457
Oscillation
  Activate, deactivate oscillation  11-459
  Asynchronized oscillation  11-456, 11-458
  Control via synchronized action  11-464
  Defining the sequence of motions  11-460
  Synchronized oscillation  11-463
  Oscillation reversal points  11-457
  Override  11-470
Parameterizable subprogram return 2-113
Parameters of the rotary axes 8-348
Parts program 13-523, 13-526
Partial infeed 11-464
Partial length 11-463
Path
  Absolute 1-73
  Relative 1-73
Path axes 5-211
Path feed 5-211
Path section 12-480
Path segmentation 12-482
Path segmentation for path axes 12-481
Path segmentation for single axes 12-482
Polynomial
  Interpolation 5-211
  Polynomial coefficient 5-205
  Polynomial definition 10-420
  Polynomial interpolation 5-204
    Denominator polynomial 5-209
Position axis 10-432
Position synchronism 13-496
Positioning movements 10-430
Power On 10-450
Preprocessing memory 9-386
Preprocessing stop 10-417
Preset offset 6-251
Program coordination 1-72
  Example 1-75
  Instructions for program coordination 1-73
Program end 1-76
Program memory 3-146
  Creating workpiece directories 3-150
  Directories 3-148
  File types 3-148
  Overview 3-147
  Programming a search path for a subprogram call 3-152
  Search path with subprogram call 3-151
  Selecting workpiece 3-151
  Workpiece directory 3-149
  Program repetition 2-117
  Program run with preprocessing memory 9-386
  Program runtime 13-528
  Programmable motion end criterion 5-229
  Programmable search path for subprogram calls 2-123
  Programming search paths for subprogram call 3-152
  Protection levels for user data 3-160
  Protection zones 4-175
    Activating/deactivating protection zones 4-180
    Contour definition of protection zones 4-178
    Define channel-specific protection zones 4-176
    Define machine-specific protection zones 4-176
    Defining protection zones 4-177
  Punching 12-476, 12-480
  Punching on 12-476
  Punching with delay Off 12-476
  Punching with delay On 12-476
  Punching, nibbling Off 12-476
  Quadrant error compensation
    Activate learning process 13-494
    Deactivate learning process 13-494
    Subsequent learning 13-494
  Quantity of parts, fixed 1-71
  READ 1-94
  Read-in disable 10-416
  Real-time variables 10-406
  Relief cut 14-543
  Relief cut elements 14-544
  REPEAT 1-69
  Repeating program sections with indirect programming CALL 2-120
  Repositioning 10-453
  Repositioning on contour 9-388
    Approach along a straight line 9-391
    Approach along quadrant 9-391
© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Advanced (PGA) – 11.02 Edition
<table>
<thead>
<tr>
<th>Approach along semi-circle</th>
<th>9-392</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach with new tool</td>
<td>9-390</td>
</tr>
<tr>
<td>Repositioning point</td>
<td>9-389</td>
</tr>
<tr>
<td>Reset</td>
<td>10-450</td>
</tr>
<tr>
<td>Resolved kinematics</td>
<td>8-346</td>
</tr>
<tr>
<td>Reversal</td>
<td></td>
</tr>
<tr>
<td>Area</td>
<td>11-464</td>
</tr>
<tr>
<td>Point</td>
<td>11-464</td>
</tr>
<tr>
<td>Rotary axes</td>
<td></td>
</tr>
<tr>
<td>Distance vectors I1, I2</td>
<td>8-346</td>
</tr>
<tr>
<td>Rotary axis</td>
<td></td>
</tr>
<tr>
<td>Direction vectors V1, V2</td>
<td>8-346</td>
</tr>
<tr>
<td>RPY angle</td>
<td>8-336</td>
</tr>
<tr>
<td>Run string as parts program line</td>
<td>1-44</td>
</tr>
<tr>
<td>Runtime response</td>
<td>1-69</td>
</tr>
</tbody>
</table>

### S

<table>
<thead>
<tr>
<th>SBBLON</th>
<th>2-126</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for character</td>
<td>1-60</td>
</tr>
<tr>
<td>Selecting a substring</td>
<td>1-62</td>
</tr>
<tr>
<td>Selection of a single character</td>
<td>1-63</td>
</tr>
<tr>
<td>Servo parameter block programmable</td>
<td>5-232</td>
</tr>
<tr>
<td>Set actual value</td>
<td>10-436</td>
</tr>
<tr>
<td>Set up variable axis transfer response</td>
<td>1-89</td>
</tr>
<tr>
<td>Setpoint coupling</td>
<td>13-495</td>
</tr>
<tr>
<td>Settable path reference</td>
<td>5-211</td>
</tr>
<tr>
<td>Setting data</td>
<td>11-458</td>
</tr>
<tr>
<td>Single axis motion</td>
<td>12-482</td>
</tr>
<tr>
<td>Single block suppression</td>
<td>2-126</td>
</tr>
<tr>
<td>Singular positions</td>
<td>7-275</td>
</tr>
<tr>
<td>Sparking-out stroke</td>
<td>11-462</td>
</tr>
<tr>
<td>Speed ratio</td>
<td>13-499</td>
</tr>
<tr>
<td>Spindle motions</td>
<td>10-437</td>
</tr>
<tr>
<td>Spindle transfer</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>1-85</td>
</tr>
<tr>
<td>RELEASE</td>
<td>1-85</td>
</tr>
<tr>
<td>Spline grouping</td>
<td>5-193</td>
</tr>
<tr>
<td>Spline interpolation</td>
<td>5-187, 5-211</td>
</tr>
<tr>
<td>A spline</td>
<td>5-188</td>
</tr>
<tr>
<td>B spline</td>
<td>5-189</td>
</tr>
<tr>
<td>C spline</td>
<td>5-190</td>
</tr>
<tr>
<td>Compressor</td>
<td>5-193</td>
</tr>
<tr>
<td>Start/stop axis</td>
<td>10-432</td>
</tr>
<tr>
<td>Station/position change</td>
<td>13-526</td>
</tr>
<tr>
<td>Status of coupling</td>
<td>9-378</td>
</tr>
<tr>
<td>Stock removal</td>
<td>14-542</td>
</tr>
<tr>
<td>Stopping and retract</td>
<td></td>
</tr>
<tr>
<td>Extended</td>
<td>13-512</td>
</tr>
<tr>
<td>String length</td>
<td>1-60</td>
</tr>
<tr>
<td>String operations</td>
<td>1-55</td>
</tr>
<tr>
<td>Structuring instruction for the Step editor</td>
<td>3-173</td>
</tr>
<tr>
<td>Subprogram call</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>1-41</td>
</tr>
<tr>
<td>Subprogram call with M/T function</td>
<td>2-136</td>
</tr>
<tr>
<td>Subprogram call, search path</td>
<td>3-151</td>
</tr>
<tr>
<td>Subprogram with path specification and parameters</td>
<td>2-122</td>
</tr>
<tr>
<td>Subprogram, external</td>
<td>2-132</td>
</tr>
<tr>
<td>Subprograms</td>
<td>2-102</td>
</tr>
<tr>
<td>Indirect subprogram call</td>
<td>2-119</td>
</tr>
<tr>
<td>Modal subprogram call</td>
<td>2-118</td>
</tr>
<tr>
<td>Nesting</td>
<td>2-103</td>
</tr>
<tr>
<td>Program repetition</td>
<td>2-117</td>
</tr>
<tr>
<td>SAVE mechanism</td>
<td>2-104</td>
</tr>
<tr>
<td>Subprogram call</td>
<td>2-109</td>
</tr>
<tr>
<td>Subprogram with parameter transfer</td>
<td>2-109</td>
</tr>
<tr>
<td>Subprograms with parameter transfer</td>
<td></td>
</tr>
<tr>
<td>Array definition</td>
<td>2-108</td>
</tr>
<tr>
<td>Parameter transfer between main program and subprogram</td>
<td>2-105</td>
</tr>
<tr>
<td>Supplementary conditions</td>
<td>1-70, 5-212, 10-450</td>
</tr>
<tr>
<td>SW limit switch</td>
<td>10-434</td>
</tr>
<tr>
<td>Switchable geometry axes</td>
<td>7-308</td>
</tr>
<tr>
<td>Synchronization run</td>
<td></td>
</tr>
<tr>
<td>Coarse</td>
<td>13-495</td>
</tr>
<tr>
<td>Fine</td>
<td>13-495</td>
</tr>
<tr>
<td>Setpoint synchronization</td>
<td>13-495</td>
</tr>
<tr>
<td>Synchronized action parameters</td>
<td>10-410</td>
</tr>
<tr>
<td>Synchronized actions</td>
<td>13-523</td>
</tr>
<tr>
<td>Static</td>
<td>9-379</td>
</tr>
<tr>
<td>Synchronized oscillation</td>
<td></td>
</tr>
<tr>
<td>Assignment of oscillating and infeed axes</td>
<td>11-465</td>
</tr>
<tr>
<td>Definition of infeed</td>
<td>11-465</td>
</tr>
<tr>
<td>Infeed in reversal area</td>
<td>11-467</td>
</tr>
<tr>
<td>Stop at reversal point</td>
<td>11-469</td>
</tr>
<tr>
<td>Synchronized action</td>
<td>11-466</td>
</tr>
</tbody>
</table>
Synchronized spindle  13-495
  Activate synchronized mode  13-501
  Block change behavior  13-500
  Coupling type  13-500
  Deactivate synchronized mode  13-501
  Define pair  13-497
  Delete coupling  13-502
  Pair  13-496
  Speed ratio  13-499
Synchronized spindle
  System variable  1-27
System variables  1-26, 13-523
  Global  13-523

T
TANG  9-353
Tangential control
  Angle limit through working area limitation  9-354
  Defining following axis and leading axis  9-353
Tangential control, activation, TANGON  9-354
Tangential control, deactivation  9-354
Technology cycles  10-445
Thread blocks  5-212
Three-digit M/G function  2-143
Tilt angle  7-270
Timer variable  10-409
Tool management  8-316
Tool monitoring, grinding-specific  8-321
Tool offset
  3D face milling  8-331
  Offset memory  8-314
  Online  8-319
Tool offsets
  Face milling  8-328
Tool orientation  7-269, 8-336
  with LEAD and TILT  7-273
Tool radius compensation, 3D  8-328
  Behavior at outside corners  8-337
  Circumferential milling  8-330, 8-331
  Insertion depth (ISD)  8-334
  Inside corners/outside corners  8-334
  Programming tool orientation  8-336
  Tool orientation  8-336
  Toolholder  8-348
  Clear/edit/read data  8-349
  Kinematics  8-346
  Torsion  13-493
  Total basic frame  6-260
  TRACYL transformation  7-290
  TRAFOOF  7-304
  Transformation TRAORI  7-268
  Transformation with a swiveling linear axis  7-267
  Transformation, 3/4-axis  7-268
  Transformation, 5-axis
    Programming in Euler angles  7-270
    Programming in RPY angles  7-271
    Programming the direction vector  7-271
  Tool orientation  7-269
  Transformation, 5-axis, face milling  7-272
  Transformation, 5-xis, programming via LEADITILT  7-269
  TRANSIT transformation  7-287
  TRAORI  7-266
  Travel to fixed stop FXS and FOCON/FOCOF  10-442
  Travel-dependent acceleration PUNCHACC  12-476, 12-477
  Traversing a contour element  14-556
  Trigger events  5-220
  Type conversion  1-56
  Type of kinematics  8-349
  Type of kinematics M  8-346
  Type of kinematics P  8-346
  Type of kinematics T  8-346

U
Uc.com, user cycles  2-139
User memory  3-153
  Data areas  3-153
  Initialization programs  3-153
  Reserved module names  3-156
Appendix 11.02

Index

V

Variable 1-26
  Arithmetic variable 1-27
  Array definition 1-34
  Assignments 1-45
  Indirect programming 1-40
  System variable 1-27
  Type conversion 1-54
  User-defined 1-26
  User-defined variable 1-29
  Variable classes 1-26
  Variable types 1-27
  Variable definition 1-29
  Variable type 1-31
  Variables
    NCK-specific global variables 1-76
  Vocabulary word 10-399

W

Wait marks 10-441
WCS 3-149
WHEN-DO 11-466
WHILE 1-68
Workpiece clamping 13-523
Workpiece counter 13-530
Workpiece directory 3-149
WPD 3-149
WRITE 1-91

Z

Zero frame 6-252
Zero offset
  Deactivating transformations 6-252
  External zero offset 6-250
  Offset using handwheel 6-249
  PRESETON 6-251
B Commands, Identifiers

- 1-41
  - 1-41
  * 1-41
  * 1-41
  / 1-41
  / 1-41
  : 1-41
  : 1-41
  + 1-41
  + 1-41
  < 1-43
  << 1-43
  <= 1-43
  <> 1-43
  =
  == 1-43
  >
  > 1-43
  >= 1-43

A

A 7-259
A1, A2 8-310, 8-312
A2 7-236
A3 7-236
A4 7-236

A5 7-236
ABS 1-41
ACC 13-453
ACOS 1-41
ACTFRAME 6-204
ALF 1-70
A_{\text{max}} 12-434
A_{\text{min}} 12-434
AND 1-44
ANZHINT 14-493, 14-495
applim 9-325
aproxLW 9-325
APW 3-139
AROTS 6-213
AS 2-122
ASIN 1-41
ASPLINE 5-155
ATAN2 1-41
AV 13-455
AX 13-446
AXCTSWE 13-480
AXIS 1-29
AXNAME 1-50, 13-446
AXSTRING 1-50, 13-446
B

B\_AND 1-45
B\_NOT 1-45
B\_OR 1-45
B\_XOR 1-45
B2 7-236
B3 7-236
B4 7-236
B5 7-236
BAUTO 5-159
BFRAME 6-203
BNAT 5-159
BOOL 1-29
BRISK 11-415
BSPLINE 5-155
BTAN 5-159
C
C2  7-236
C3  7-236
C4  7-236
C5  7-236
C  CAC  5-154
C  CACN  5-154
C  CACP  5-154
C  CALCDAT  14-490, 14-505
C  CALL  2-107
C  CANCEL  10-354
C  CASE  1-58
C  CDC  5-154
C  CFINE  6-214
C  CHANDATA  3-134
C  CHAR  1-29
C  CHKDNO  8-306
C  CIC  5-154
C  CLEARM  1-67
C  CLRINT  1-70
C  CMISS  6-207
C  COARSE  13-450, 13-454, 13-455
C  COARSEA  5-197
C  COMPLETE  3-132, 3-133
C  COMPOF  5-169, 5-179
C  COMPON  5-179, 9-342
C  CONTDCON  14-498
C  CONTPRON  14-490, 14-491, 14-503, 14-504
C  COS  1-41
C  COUPDEF  13-450, 13-452, 13-454
C  COU  DEL  13-450, 13-452, 13-457
C  COUPOF  13-450, 13-456, 13-457
C  COUPON  13-450, 13-456, 13-457
C  COUPRES  13-450, 13-457
C  CP  7-245
C  CROTH  4-148
C  CPROTDEF  4-144, 4-146
C  CROT  6-207
C  CROTS  6-213
C  CSCALE  6-207
C  CSPLINE  5-155
C  CTAB  9-325
C  CTABDEF  9-325

CTABDEL  9-325
CTABEND  9-325
CTABINV  9-325
CTRANS  6-207
CUT3DC  8-292
CUT3DF  8-292
CUT3DFF  8-292
CUT3DFS  8-292
CUTCONOF  8-289
CUTCONON  8-289

D
D  DEF  1-29
D  DEFAULT  1-58
D  DEFINE  2-122
D  DELTGT  5-194
D  DELT  8-280
D  DISABLE  1-70
D  DISPLOF  2-109
D  DISPR  9-346
D  DIV  1-41
D  DO  10-354, 11-421
D  DRFOF  6-218
D  DUPLO_NR  8-280
D  DV  13-455
D  DZERO  8-309

E
E  EAUTO  5-159
E  ELSE  1-60
E  ENABLE  1-70
E  ENAT  5-159
E  ENDFOR  1-60
E  ENDF  1-60
E  ENDO  LOOP  1-60
E  ENDP  OS  11-421
E  ENDP  ROC  10-384
E  ENDO  HILE  1-60
E  ERG  14-505
E  ERROR  14-491, 14-498
E  ETAN  5-159
E  EVERY  10-354
EXECTAB  14-504
EXECUTE  4-144, 4-146, 14-491, 14-498
EXP   1-41
EXTCALL  2-113
EXTERN  2-101

F
FA  11-418, 13-453
FALSE  1-25
FCTDEF  8-283
FCUB  9-339
FINE  13-450, 13-455
FINEA  5-197
FLIN  9-339
FMA  15-517
FNORM  9-339
FOR  1-60
FPO  9-339
FRAME  1-29
FRC  15-518
FRCM  15-518
FROM  10-354
FS  13-450
FTOC  8-283
FTOCOF  8-283
FTOCON  8-283
FW  9-325

G
G05  7-263
G07  7-263
G1  11-415
G153  6-218
G25,G26  9-318
G4  11-417
GEOAX  7-271
GET  1-78
GETACTTD  8-308
GETD  1-78
GETDNO  8-307
GETSELT  8-280
GETT  8-280
GOTOB  1-58
GOTOF  1-58
GUD  3-128, 3-132, 3-137, 3-139
I
I1,I2  8-310
ID  10-353
IDS  10-353
IF  1-60
IF-ELSE-ENDIF  1-60
IFRAME  6-204
II1,II2  11-422
INDEX  1-53
INIT  1-66
INITIAL  3-133
INT  1-29
INTERSEC  14-490, 14-503
IPOENDA  5-197
IPOSTOP  13-450, 13-453, 13-455
ISAXIS  13-446
ISD  8-292, 8-298
ISNUMBER  1-50

K
KТАB  14-493, 14-495, 14-501, 14-504

L
LEAD  7-236, 8-300
LEADOF  9-333
LEADON  9-333
LIFTFAST  1-70
LOCK  10-354
LOOP  1-60
LOOP-ENDLOOP  1-61
LS  13-450
LW  9-325

M
M  8-312
M17  2-97
MATCH  1-53
<table>
<thead>
<tr>
<th>Command/Identifier</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCALL</td>
<td>2-106</td>
</tr>
<tr>
<td>MEAC</td>
<td>5-186, 5-194</td>
</tr>
<tr>
<td>MEAFRAME</td>
<td>6-220</td>
</tr>
<tr>
<td>MEAS</td>
<td>5-183</td>
</tr>
<tr>
<td>MEASA</td>
<td>5-186</td>
</tr>
<tr>
<td>MEAW</td>
<td>5-183</td>
</tr>
<tr>
<td>MEAWA</td>
<td>5-186</td>
</tr>
<tr>
<td>MI</td>
<td>6-209</td>
</tr>
<tr>
<td>MIRROR</td>
<td>6-204</td>
</tr>
<tr>
<td>MMC</td>
<td>13-486</td>
</tr>
<tr>
<td>MOD</td>
<td>1-41</td>
</tr>
<tr>
<td>MOV</td>
<td>10-390</td>
</tr>
<tr>
<td>MPF</td>
<td>3-128</td>
</tr>
<tr>
<td>MU</td>
<td>7-261</td>
</tr>
<tr>
<td>MZ</td>
<td>7-261</td>
</tr>
<tr>
<td>NEWT</td>
<td>8-280</td>
</tr>
<tr>
<td>Nibbling</td>
<td>12-438</td>
</tr>
<tr>
<td>NN</td>
<td>14-491</td>
</tr>
<tr>
<td>NO.</td>
<td>14-505</td>
</tr>
<tr>
<td>NOC</td>
<td>13-455</td>
</tr>
<tr>
<td>NOT</td>
<td>1-44</td>
</tr>
<tr>
<td>NPROT</td>
<td>4-148</td>
</tr>
<tr>
<td>NPROTDEF</td>
<td>4-144, 4-146</td>
</tr>
<tr>
<td>NUMBER</td>
<td>1-50</td>
</tr>
<tr>
<td>OEMIPO1/2</td>
<td>5-196</td>
</tr>
<tr>
<td>OF</td>
<td>1-59</td>
</tr>
<tr>
<td>OFFN</td>
<td>7-252, 7-253</td>
</tr>
<tr>
<td>OR</td>
<td>1-44</td>
</tr>
<tr>
<td>ORIC</td>
<td>8-300</td>
</tr>
<tr>
<td>ORID</td>
<td>8-300</td>
</tr>
<tr>
<td>ORIMCS</td>
<td>8-300</td>
</tr>
<tr>
<td>ORIMKS</td>
<td>7-240, 7-242</td>
</tr>
<tr>
<td>ORIS</td>
<td>8-300</td>
</tr>
<tr>
<td>ORIWCS</td>
<td>8-300</td>
</tr>
<tr>
<td>ORIWKS</td>
<td>7-240, 7-242</td>
</tr>
<tr>
<td>OS</td>
<td>11-414, 11-417</td>
</tr>
<tr>
<td>OSC</td>
<td>8-300</td>
</tr>
<tr>
<td>OSCILL</td>
<td>11-421, 11-423</td>
</tr>
<tr>
<td>OSCTRL</td>
<td>11-414, 11-418</td>
</tr>
<tr>
<td>OSE</td>
<td>11-414, 11-418</td>
</tr>
<tr>
<td>OSNOC</td>
<td>11-414, 11-421</td>
</tr>
<tr>
<td>OSO2</td>
<td>11-414</td>
</tr>
<tr>
<td>OSOF</td>
<td>8-300</td>
</tr>
<tr>
<td>OSP</td>
<td>11-415</td>
</tr>
<tr>
<td>OSP1</td>
<td>11-414, 11-421</td>
</tr>
<tr>
<td>OSP2</td>
<td>11-421</td>
</tr>
<tr>
<td>OSS</td>
<td>8-300</td>
</tr>
<tr>
<td>OSSE</td>
<td>8-300</td>
</tr>
<tr>
<td>OST</td>
<td>11-417</td>
</tr>
<tr>
<td>OST1</td>
<td>11-414, 11-421</td>
</tr>
<tr>
<td>OST2</td>
<td>11-414, 11-421</td>
</tr>
<tr>
<td>OVRA</td>
<td>13-453</td>
</tr>
<tr>
<td>PDELAYOF</td>
<td>12-434</td>
</tr>
<tr>
<td>PDELAYON</td>
<td>12-434</td>
</tr>
<tr>
<td>PFRAME</td>
<td>6-204</td>
</tr>
<tr>
<td>PKT</td>
<td>14-505</td>
</tr>
<tr>
<td>PL</td>
<td>5-158, 5-175</td>
</tr>
<tr>
<td>PO</td>
<td>5-175</td>
</tr>
<tr>
<td>POLY</td>
<td>5-175</td>
</tr>
<tr>
<td>POLYNOMIAL</td>
<td>14-492, 14-499</td>
</tr>
<tr>
<td>POLYPATH</td>
<td>5-175</td>
</tr>
<tr>
<td>PON</td>
<td>12-434, 12-440</td>
</tr>
<tr>
<td>PONS</td>
<td>12-434</td>
</tr>
<tr>
<td>POS</td>
<td>13-456</td>
</tr>
<tr>
<td>POSP</td>
<td>11-421</td>
</tr>
<tr>
<td>POT</td>
<td>1-41</td>
</tr>
<tr>
<td>PRESETON</td>
<td>6-217, 6-220</td>
</tr>
<tr>
<td>PRIO</td>
<td>1-70</td>
</tr>
<tr>
<td>PROC</td>
<td>2-97</td>
</tr>
<tr>
<td>PUNCHACC</td>
<td>12-434</td>
</tr>
<tr>
<td>PUTFTOC</td>
<td>8-283</td>
</tr>
<tr>
<td>PUTFTOCF</td>
<td>8-283</td>
</tr>
<tr>
<td>PW</td>
<td>5-157</td>
</tr>
<tr>
<td>QEC</td>
<td>13-448</td>
</tr>
<tr>
<td>QECODAT.MPF</td>
<td>13-449</td>
</tr>
<tr>
<td>QECLRN.SPF</td>
<td>13-449</td>
</tr>
</tbody>
</table>
QECLRNOF 13-448
QECLRNON 13-448
QECTEST.MPF 13-449

S
S1,S2 13-452, 13-457
SAVE 1-71, 2-96
SBLOF 2-110
SBLON 2-110
SC 6-209
SCPARA 5-198
SD 5-157
SET 1-34
SETDNO 8-307
SETINT 1-70
SETM 1-67
SETPIECE 8-280
SIN 1-41
S_{\text{max}} 12-434
S_{\text{min}} 12-434
SOFT 11-415
SON 12-434, 12-439, 12-440
SONS 12-434
SPI 13-446, 13-453
SPIF1 15-530
SPIF2 15-530
SPLINE 14-492, 14-499
SPLINEPATH 5-161
SPN 12-438
SPOF 12-434
SPOS 13-453
SPP 12-438
SQRT 1-41
SR 15-531
SRA 15-531
ST 15-531
START 1-66
STARTFIFO 9-344
STOPFIFO 9-344
STOPRE 5-183, 5-190, 5-192, 9-344, 11-416
STOPREOF 10-375
STRING 1-29
STRINGFELD 1-48
STRINGVAR 1-48
STRLEN 1-53
SUPA 6-218
SYNFCT 10-381
SYNR 3-137
SYNRW 3-137
T
TABNAME 14-491, 14-498, 14-502, 14-504
TAN 1-41
TANG 9-316
TANGOF 9-316
TANGON 9-316
TE 5-186
THREAD 14-492, 14-499
TILT 7-236, 8-300
TLIFT 9-316

© Siemens AG, 2002. All rights reserved
SINUMERIK 840D/840D/810D Programming Guide Advanced (PGA) – 11.02 Edition
A-723
Appendix
Commands, Identifiers

TOLOWER 1-52
TOUPPER 1-52
TR 6-209
TRAANG 7-253, 7-259
TRACYL 7-250, 7-253
TRAFOOF 7-232, 7-250, 7-253, 7-259, 7-267
TRAILOF 9-321
TRAILON 9-321
TRANSMIT 7-250
TRAORI 7-234
TRUE 1-25
TRUNC 1-41

U
U1,U2 11-422
UNLOCK 10-354
UNTIL 1-60, 1-62

V
V1,V2 8-310
VAR 2-99
VARIB 14-502, 14-505

W
WAIT 1-67
WAITC 13-450, 13-453
WAITE 1-67
WAITM 1-66
WAITMC 1-67
WAITP 11-417
WALIMON 9-318
WCS 11-428
WHEN 10-354
WHEN-DO 11-421
WHENEVER 10-354
WHENEVER-DO 11-421, 11-424
WHILE 1-60
WZ 8-280

TOLOWER 1-52
TOUPPER 1-52
TR 6-209
TRAANG 7-253, 7-259
TRACYL 7-250, 7-253
TRAFOOF 7-232, 7-250, 7-253, 7-259, 7-267
TRAILOF 9-321
TRAILON 9-321
TRANSMIT 7-250
TRAORI 7-234
TRUE 1-25
TRUNC 1-41

U
U1,U2 11-422
UNLOCK 10-354
UNTIL 1-60, 1-62

V
V1,V2 8-310
VAR 2-99
VARIB 14-502, 14-505

W
WAIT 1-67
WAITC 13-450, 13-453
WAITE 1-67
WAITM 1-66
WAITMC 1-67
WAITP 11-417
WALIMON 9-318
WCS 11-428
WHEN 10-354
WHEN-DO 11-421
WHENEVER 10-354
WHENEVER-DO 11-421, 11-424
WHILE 1-60
WZ 8-280
To
SIEMENS AG
A&D MC BMS
P.O. Box 3180
D-91050 Erlangen, Germany
Phone: ++49-(0)180-5050-222  [Hotline]
Fax: ++49-(0)9131-98-2176  [Documentation]
Email: motioncontrol.docu@erlf.siemens.de

Suggestions and/or corrections

Suggestions

Corrections

For publication/manual:

SINUMERIK 840D/840Di/810D
Programming Guide
Advanced

User Documentation

From

Order No.: 6FC5298-6AB10-0BP2
Edition: 11.02

Should you come across any printing errors when reading this publication, please notify us on this sheet. Suggestions for improvement are also welcome.
Overview of SINUMERIK 840D/840Di/810D Documentation (11.2002)

General Documentation

- SINUMERIK 840D/810D
  - Brochure

- SINUMERIK 840D/840Di/810D
  - Catalog Ordering Info. NC 60 *

- SINUMERIK SIMODRIVE
  - Accessories

User Documentation

- SINUMERIK 840D/840Di/810D
  - AutoTurn
  - Operator’s Guide
  - Diagnostics Guide *)

- SINUMERIK 840D/840Di/810D
  - System Overview
  - Operator’s Guide
  - Manual’s Guide

- SINUMERIK 840D/840Di/810D
  - Configuring (HW) *)
  - Operator Components (HW) *)

Manufacturer/Service Documentation

- SINUMERIK 840D/840Di/810D
  - Description of Functions
  - Drive Functions *)

- SINUMERIK 840D/840Di/810D
  - Description of Functions
  - Tool Management

- SINUMERIK 840D/840Di/810D
  - Configuring Kit
  - HMI Embedded

- SINUMERIK 840D/840Di/810D
  - IT Solutions

- SINUMERIK 840D/840Di/810D
  - Description of Functions
  - Linear Motor

Electronic Documentation

- SINUMERIK 840D/840Di/810D
  - DOC ON CD *)

* These documents are a minimum requirement